VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Komplexní monitoring infrastruktury

Comprehensive Infrastructure Monitoring

Bc. Lukáš Klobasa

Diplomová práce

Vedoucí práce: Ing. Lukáš Vojáček, Ph.D.

Ostrava, 2021

Abstrakt

Tato práce se zaměřuje na návrh a tvorbu víceúrovňového monitoringu infrastruktury, která má své využití v praxi. Ukazuje důležitost a využití monitorovacího systému a jeho přehledné zobrazení s možností procházení historických dat a sloučení několika různých zdrojů do jednoho graficky přívětivého prostředí. Obsahuje také srovnání a výběr technologií pro monitorování infrastruktury. Dále zahrnuje popis jednotlivých technologií použitých na různých úrovních infrastruktury a jejich potřeby pro monitorovací systém. Je zde popsán proces tvorby grafů a vizálních prvků pro monitorvací systém a také vytváření poplachových kritérií. Rovněž je zde vyobrazen proces zálohování a obnovy na základě pravidel pro zálohování. V práci je vidět výsledný návrh infrastruktury, proces její tvorby, vývoje monitorovacího systému a ukazuje jak je možné v dnešní době vytvořit infrastrukturu za pomocí nejnovějších technologií.

Klíčová slova

Monitorování, infrastruktura, grafické rozhraní, poplachové kritéria, vývoj

Abstract

This masters thesis focuses on design and creation of multilevel infrastructure monitoring system, which has its use in practice. It shows the importance and usability of the monitoring system and its clear display with the possibility of browsing historical data and merging several different sources into one graphically user friendly interface. It also contains a comparison and selection of technologies for infrastructure monitoring. Description of the individual technologies used at different levels of the infrastructure and their needs for the monitoring system is also included. Next The process of creating graphs and visual elements for the monitoring system is described here, as well as the creation of alerts. It also shows the backup and recovery process based on the best practices. The work shows the final infrastructure design, the process of its creation, the development of the monitoring system and also how it is possible to create an infrastructure using the latest technologies today.

Keywords

Monitoring, infrastructure, graphical interface, alerts, development

Poděkování

Rád bych na tomto místě poděkoval mému vedoucímu práce Ing. Lukáši Vojáčkovi, Ph.D. za konzultace, poznatky a přátelský přístup při tvorbě této práce. Dále bych také rád poděkoval mému kolegovi Ing. Martinu Golasowskému, Ph.D. za rozšíření oblastí a zaučení nových technologií na IT4Innovations a za kontrolu mé práce. Nakonec bych rád poděkoval své rodině za podporu během celého studia a mému dědovi za korekturu gramatiky textu.

Obsah

Seznam použitých zkratek a symbolů

ELK	–	ElasticSearch, Logstash, Kibana
JSON	–	JavaScript Object Notation
REST	–	Representational State Transfer
GUI	–	Graphical user interface
API	–	Application Programming Interface
SNMP	–	Simple Network Management Protocol
GPL	–	General Public License
YAML	–	YAML Ain't Markup Language
OSD	–	Object-based storage
YUM	–	Yellowdog Updater, Modified
AD	–	Active Directory
LDAP	–	Lightweight Directory Access Protocol
VPN	–	Virtual Private Network
CPU	–	Central Processing Unit
SSH	–	Secure Shell
RBD	–	RADOS Block Device

Seznam obrázků

Kapitola 1

Úvod

Cílem této práce bylo vytvoření systému monitoringu víceúrovňové infrastruktury, který se využije v praxi. V této práci je vidět důležitost monitoringu a sběr jednotlivých metrik z různých datových zdrojů a implementace technologií pro dosažení komplexního systému monitorování všech úrovní infrastruktury. Při vytváření této práce byl kladen důraz na poskytování služeb infrastruktury a všech jejch částí. Z tohoto důvodu byl také důležutý výběr technologií pro monitorování. Dále bylo důležité vybrat vhodně hlavní nástroj pro zobrazení jednotlivých metrik a tvorbu dashboardů, který by umožňoval snadnou manipulaci, přehledné grafické rozhraní a možnost připojení více zdrojů dat. Při vytváření monitorovacího systému víceúrovňové infrastruktury je důležité sloučit více různých úrovní do jednoho monitorovacího systému a toho se v této práci také snažilo dosáhnout. Snaha byla také zajistit co největší bezpečnost.

První a druhá kapitola obsahuje úvod do monitoringu a proč vlastně monitoring využívat. Ve třetí kapitole je již samotný popis infrastruktury, její jednotlivé úrovně a jaké její části by bylo třeba monitorovat. Ve čtvrté kapitole je popsán rozbor jednotlivých monitorovacích nástrojů a jejich výběr s ohledem na potřeby infrastruktury. Pátá kapitola obsahuje návrh schématu monitoringu a technologie, které byly využity pro monitorování všech úrovní infrastruktury. V páté kapitole je popsána již samotná implementace monitorovacího systému a skriptů pro sběr jednotlivých metrik. Dále je zde popsána tvorba dashboardů a grafické zobrazení sbíraných metrik. Nakonec je v páté kapitole popsána tvorba alertů a jejich připojení ke komunikačním kanálům. V poslední šesté kapitole je popsáno řešení zálohy a proces jejího případné obnovení.

Kapitola 2

Monitoring

Pro uvedení čtenářů, kteří nemají zkušenosti s monitorováním systémů výpočetní infrastruktury, považuji za nutné vysvětlit co vlastně právě monitoring je. V následující podkapitole je popsán obecný proces návrhu monitorovacího systému, jeho možnosti, výhody, složení a také proč vlastně takový monitorovací systém v dnešní době používat.

2.1 Proč využívat monitoring

Následující poznatky a informace vychází z knihy s názvem Effective monitoring and alerting [1]. Doba, kdy bylo možné vysledovat selhání nějakého procesu podle několika možných příčin, je dávno pryč. Dnešní informační systémy se staly tak složitými, že efektivní hledání jakýchkoliv problémů, vyžaduje vysoký výkon, mnoho dovedností a hluboké porozumění datové interpretace. Standardy dostupnosti se v odvětví informačních technologií drží na vysoké úrovni a posouvají se ještě dále. Informační systémy musí být vybaveny výkonným hardwarem, protože nedostatek informací a rychlosti by mohl znamenat značnou ztrátu času a v některých případech také ztrátu příjmů.

Z tohoto důvodu se v dnešní době používají monitorovací systémy. Monitoring umožňuje administrátorům vyřešit různé komplikace, ještě před tím, než se stanou opravdovými problémy. Umožňuje udržet vysokou dostupnost a také poskytovat vysokou kvalitu služeb. Na základě historických dat umožňuje úpravy a predikci trendů. Monitoring je proces dohledu nad tokem dat v systému. Cílem monitoringu je identifikovat poruchy a pomáhat při jejich následném odstraňování. Techniky používané při monitorování informačních systémů protínají oblasti jako je zpracování dat v reálném čase, statistka a analýza dat. Sada softwarových komponent, které se využívají pro sběr dat, jejich zpracování a prezentaci, se nazývá monitorovací systém.

Pojem monitorování silně závisí na kontextu. Obecně se jedná o proces uvědomění si stavu systému. Děje se to dvěma způsoby, proaktivní a reaktivní. První způsob zahrnuje sledování vizuálních indikátorů, jako jsou například různé dashboardy, timeseries grafy (grafy časových řad), a samotné panely, které zobrazují konkrétní data. Druhým způsobem je myšleno automatizované doručení

oznámení o chybě samotným administrátorům s cílem upozornit je na to, že v systému není něco v pořádku. Toto oznámení může být provedeno přes různé komunikační kanály. Obvykle se tomuto způsobu říká alerting.

Alerting je schopnost monitorovacího systému detekovat chyby, případně jakékoliv změny v systému, které by systém a jeho funkčnost či dostupnost mohly ovlivnit. Alerting má za účel v případě chyby informovat administrátory například přes e-mail nebo SMS a dát jim vědět o tom že se v systému něco děje a bude možná nutné danou komplikaci řešit. Jelikož je tento proces automatizovaný, administrátoři nemusí být neustále aktivní a nemusí sledovat grafy v reálném čase. Právě rychlá detekce komplikací systému a varování je jedním z nejdůležitějších prvků monitorovacího systému. Problém ale spočívá v přesném nastavení jednotlivých prahů hodnot, při kterých nastává zmíněné varování a odeslání upozornění. Nechceme, aby jsme zbytečně dostávali upozornění při jakémkoli malém výkyvu, který by nijak zásadně systém neovlivnil. Z tohoto důvodu je velmi důležité manuální nastavení těchto alertů a na základě delšího pozorování stanovit přesné hranice pro každou instanci zvlášť.

Ve firemním prostředí a v rámci poskytování služeb je dostupnost všech prvků a instancí klíčová. Jakákoliv odstávka nebo dokonce i částečná nedostupnost služeb uživatelům může znamenat značné finanční ztráty. Z tohoto pohledu je také velmi důležité využívat právě monitorovací systémy. Monitoring je důležitý při řešení incidentů, kdy je možno identifikovat přesný čas a místo, kde chyba nastala. Díky této informaci můžeme najít konkrétní logy a jasně identifikovat příčinu chyby. Dostupnost historických dat nám pak umožňuje například identifikovat výkonnostní trendy a optimalizovat provoz infrastruktury.

2.2 Složení monitorovacího systému

Monitorovací systém se skládá ze tří hlavních prvků:

1. Sběr dat

   Data, která se využívají pro monitorování, jsou shromažďována agenty z různých hardwarových zařízení, serverů, databází a také samotných instancí služeb. Zdroje dat jsou například logy, statistiky o zařízení nebo systémové informace. Tito agenti jednotlivé data shlukují, přidávají jim popisky a vytváří z nich metriky, které se využívají pro zobrazení v čase. Tyto metriky se následně ukládají do databáze.

   Metriky, které jsou tvořeny agenty, jsou kolekce číselných datových vstupů nebo kvantifikovatelných kategorií, které jsou uspořádané do skupin chronologicky po sobě jdoucích seznamů. Každý jednotlivý datový vstup je složen ze zaznamenané hodnoty, časového razítka, které označuje, kdy k zaznamenání došlo a sady vlastností, které hodnotu popisují.

   Sběr dat může být kontinuální nebo může probíhat v časových intervalech. Agenty, kteří jednotlivé data sbírají, je nutné nasadit na všechny instance, které je třeba monitorovat. Pro

sběr dat existují dva přístupy. Agenti mohou data odesílat na centrální server na základně push přístupu. Druhý přístup je pull, kdy dochází ke stahování metrik z agentů prostřednictvím centrálního procesu.

2. Agregace a ukládání dat

   Data jsou uložena do databáze a jsou roztříděna dle jejich vlastností. Tato data se sumarizují a vytváří jednotlivé timeseries, ze kterých jsou následně tvořeny real time grafy. Velkou výhodou timeseries databází je také rychlé vyhledávání pomocí časového intervalu.

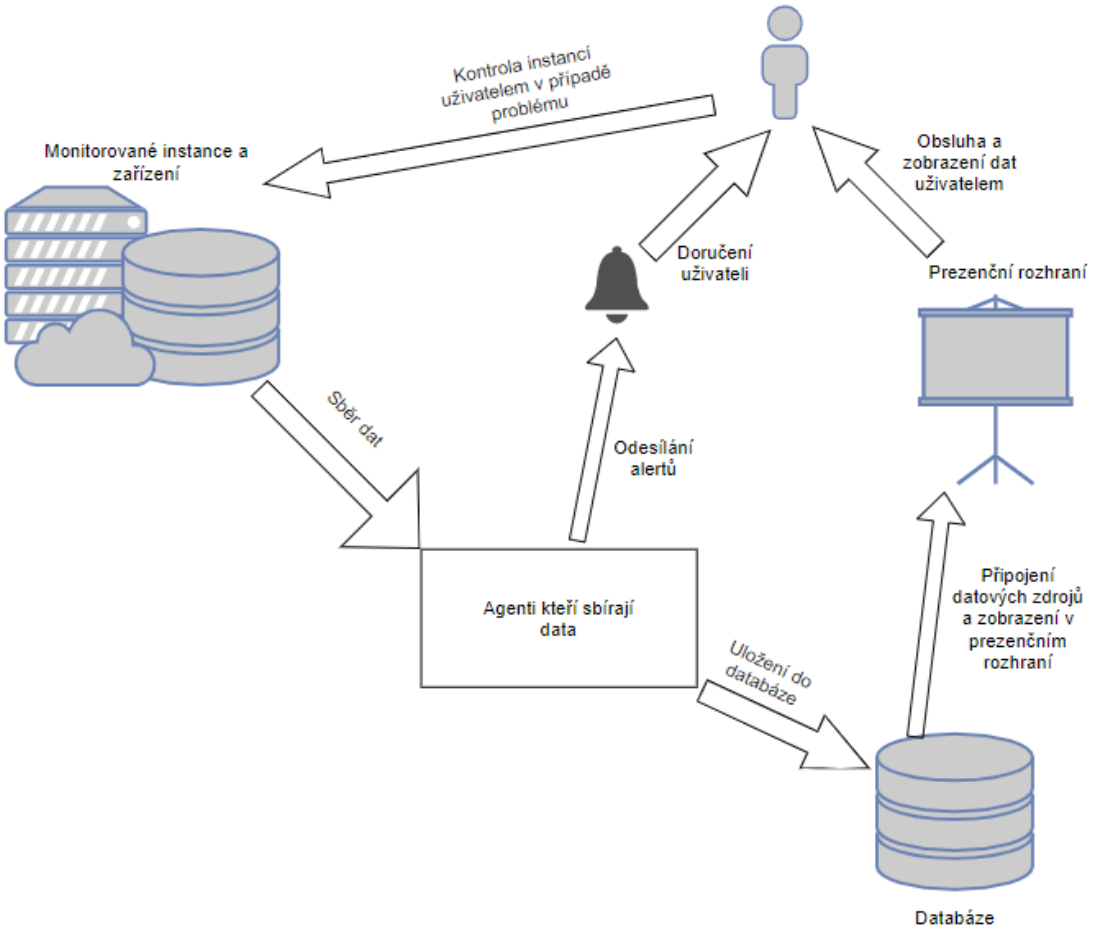
   Timeseries obsahují dvourozměrné data, kde jsou uloženy konkrétní zaznamenané hodnoty a čas. Z toho vyplývá, že všechna data v timeseries databázi obsahují časové razítko. Díky tomu, že můžeme zobrazovat konkrétní metriky z různých zdrojů pro stejný časový interval, je možné vykreslovat do jednoho grafu.

   Na základě těchto timeseries se tvoří jednotlivé aletry a jejich thresholdy. Pokud je překročena daná hranice, je spuštěn alert a dochází k odeslání oznámení přes nastavené komunikační kanály.

3. Prezentace dat

   Uložená a správně roztříděná data je možné zobrazit pomocí uživatelsky přívětivého rozhraní a tvořit jednotlivé timeseries grafy, statistiky a panely pro dané metriky. Pomocí dotazů do databáze následně tvoříme samotnou prezentaci. V prezenční vrstvě lze poté vidět souhrnný přehled různých dat a je možné data v čase procházet a nastavovat si určitá časová rozmezí. Jednotlivé panely je možné také graficky upravit, aby zjištěná chyba byla barevně zvýrazněna. Každý monitorovací systém má své vlastní rozhraní a konfiguraci.

Na obrázku 2.1 je vidět schéma monitorovacího systému a jeho interakci mezi jednotlivými prvky a uživatelem.



Obrázek 2.1: Monitorovací systém

Kapitola 3

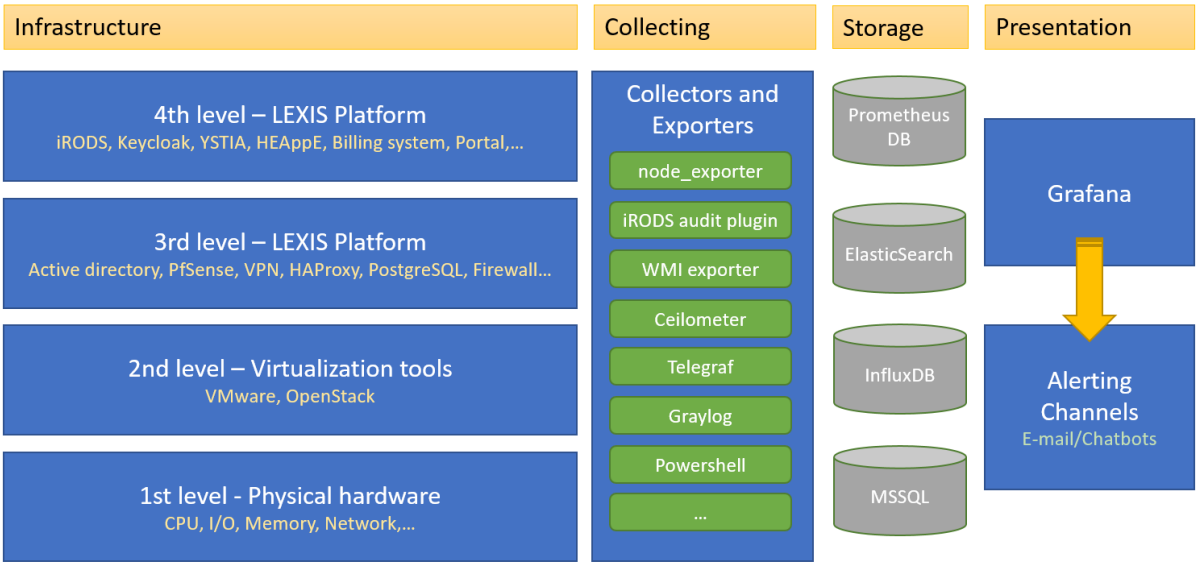
Popis Infrastruktury

Infrastruktura jejíž monitoring je předmětem této práce je aktuálně používaná v praxi a je plně funkční. Tato infrastruktura je vytvořena pro plnění výzkumných úkolů v rámci projektu H2020 projektu Lexis viz [2] na IT4Innovations. Infrastruktura je určena pro kompletní správu instancí pomocí cloudového řešení, poskytování úložiště CEPH, akcelerovaných datových uzlů a propojení s HPC clustery na IT4Innovations. Nároky byly kladeny na snadnou rozšiřitelnost a správu virtuálních PC, ať už se systémem Linux, jako například CentOS7 a Ubuntu nebo také Windows jako například edice Windows server a Windows 10. Pro virtualizaci jsou použity nástroje VMware a OpenStack, které jsou popsány dále v práci. Na těchto dvou virtualizačních platformách jsou nasazeny potřebné služby pro kompletní běh infrastruktury. Vzhledem k tomu, že infrastruktura je použita v praxi a výpadek jakékoliv služby či instance by mohl znamenat omezení partnerů projektu v řešení jejich úkolů, je nutné nedostupnosti jakýchkoliv služeb zabránit a eliminovat případné výpadky na co nejkratší čas. Z tohoto důvodu je tedy nutné monitorovat všechny části dané infrastruktury. Proto bylo třeba implementovat jednotný systém, ve kterém by bylo možné monitorovat všechny části na jednom místě a na základě rešerše monitorovacích systému vybrat ten nejvhodnější. Infrastruktura byla rozdělena do několika úrovní, které jsou pospány níže. Pro zachování bezpečnosti infrastruktury jsou některé detaily záměrně vynechány.

3.1 Úrovně infrastruktury

Infrastruktura je rozdělena na několik úrovní, které je třeba monitorovat. První úrovní je ta nejnižší, a to hardwarová úroveň. Zde je třeba monitorovat jednotlivé statistiky zařízení, na kterých běží dané virtualizační nástroje, úložiště a síť. Další úrovní je druhý stupeň, kde jsou virtualizační nástroje, které zajišťují základ infrastruktury a běží na nich virtualizované instance jak pro chod infrastruktury, které jsou součástí třetí úrovně, tak také samotné instance externích pracovníků infrastruktury, které jsou zařazeny do čtvrté úrovně. Třetí úrovní jsou myšleny nástroje pro chod infrastruktury a její správu, jako například DHCP, active directory a vpn servery. Čtvrtá úroveň

jsou již samotné instance, které jsou zde vytvořeny a běží na nich konkrétní služby pro potřeby zákazníků dané infrastruktury. Jednotlivé úrovně, jejich obsah a celkový diagram pro monitorování je vidět na obrázku 3.1. Jednotlivé úrovně jsou popsány dále v práci.



Obrázek 3.1: Úrovně infrastruktury

3.2 První úroveň

První úroveň infrastruktury se skládá z hardware, na kterém běží veškeré služby infrastruktury.

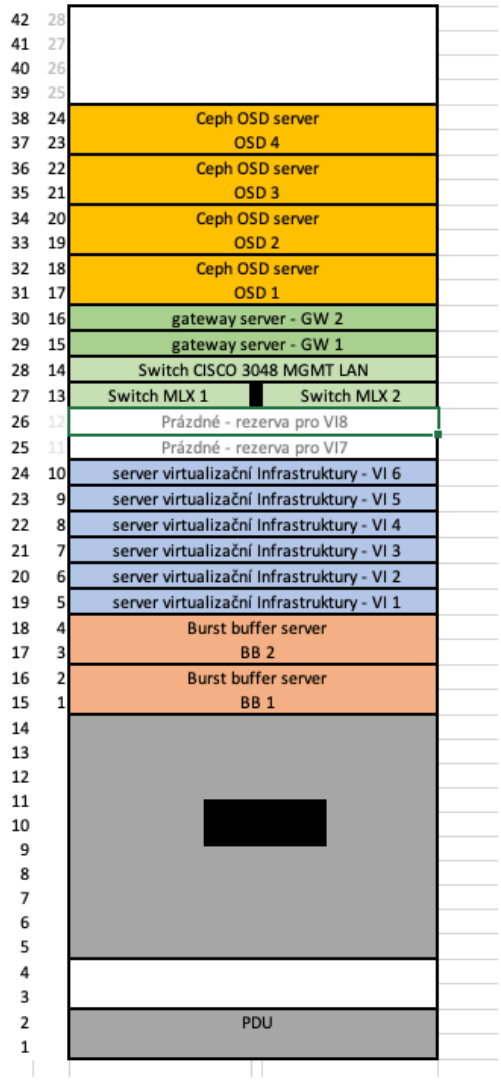
Hardware je zasazen do racku, který se skládá z několika hardwarových prvků. Úložiště infrastruktury je poskytováno prostřednictvím software CEPH, o kterém se zmiňuji dále v práci viz 3.3.1. V našem případě využíváme pro CEPH úložistě, které se skládá ze čtyř OSD serverů o celkové kapacitě 120 TB v HDD a 32 TB v SSD. Disky jsou připojeny do switchů pomocí 25 Gbit/s Ethernet sítě.

Pro síť jsou zde dva switche o rychlosti 100 Gbit/s a jeden 100 Gbit/s ethernet switch pro management LAN síť. Cisco 3048 switch využívá Cisco NX-OS operační systém, obsahuje 48 ethernetových portů a je určený pro zákazníky vyžadující gigabitové připojení. Také pro síť zde jsou dva gateway servery, které jsou pomocí Infiniband propojené s HPC clustery a pomocí 100GB/s mellanox sítě připojeny do switchů.

Dále je v racku šest virtualizačních serverů, které využívají procesory s mikroarchitekturou Intel Cascade Lake, které mají 40 jader a 192 GB RAM. Tyto servery jsou zapojeny do switchů pomocí 100 Gbit/s Mellanox sítě. Na těchto virtualizačních serverech běží služby pro cloud, VMware a OpenStack. Dále jsou zde dva Burst Buffer servery Intel Xeon Gold se 40 jádry, osazené 12.8 TB

NVMe SSD disků značky Intel a 512GB paměti typu NVDIMM. Ty jsou opět připojeny pomocí 100 Gbit/s Mellanox sítě. Na obrázku 3.2 je vidět schéma osazeného racku.

Jednotlivé metriky, které je třeba monitorovat na zmíněném hardwaru, jsou vytížení hardwarových zařízení, jako jsou například RAM, CPU, vytížení sítě, TCP a UDP metriky. Servery obsahují IPMI rozhraní, které využíváme. Dále je třeba také monitorovat vytížení disků, kapacitu a celkový stav CEPH úložiště. Tyto metriky pak je třeba také vizualizovat pro snadný přehled o stavu jednotlivých zařízení.



Obrázek 3.2: Schéma osazeného racku

3.3 Druhá úroveň

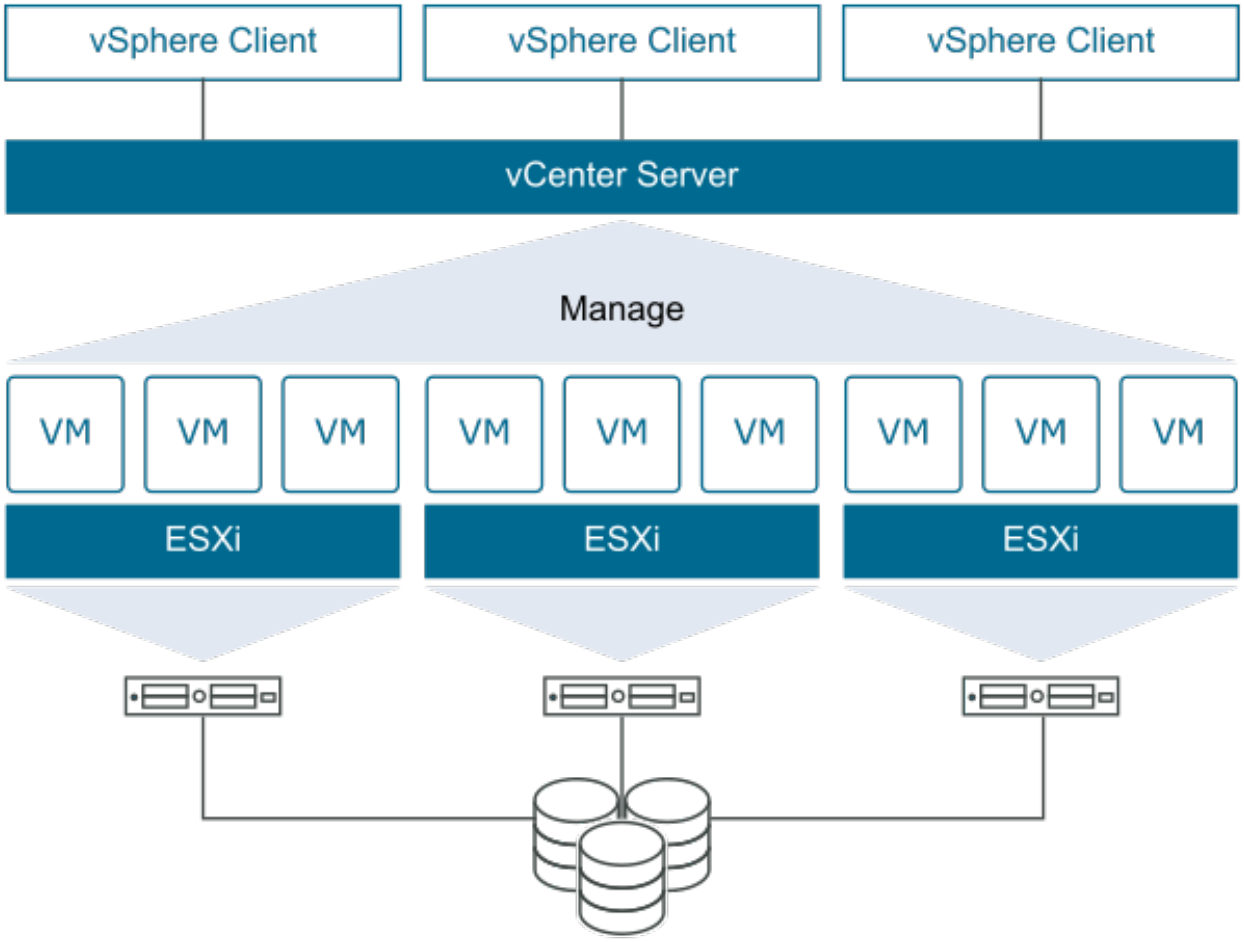
Druhá úroveň infrastruktury obsahuje software a nástroje, které jsou určeny pro chod infrastruktury a její řízení. Jedná se o řízení úložiště pomocí nástroje CEPH a virtualizační nástroje VMware a Openstack pro cloudové řešení a tvorbu instancí. Níže je přehled jednotlivých nástrojů.

3.3.1 CEPH

CEPH úložiště je open source projekt, který poskytuje softwarově definované řešení úložiště. Je to distribuovaný systém pro úložiště, který je masivně škálovatelný a vysoce výkonný. Od základu tento systém byl navržen, aby byl škálovatelný až do úrovně exabytů. V dnešní době většina veřejných, privátních a hybridních cloudových modelů spoléhá právě na budování obrovských infrastruktur a tedy na vysokou škálovatelnost a výkon a z tohoto důvodu se CEPH stal velmi využívaný pro správu a kompletní řešení cloudového úložiště. Díky tomu, že CEPH je open source jedná se taky o řešení, které není náročné na náklady. Dále je také velkou výhodou, že se CEPH dá nasadit na běžně dostupný hardware a nejsou tedy potřeba speciální řadiče jako SAS, SAN a podobně. Obsahuje také mnoho funkcí a je stále aktualizován a vyvíjen. Základ CEPHu je postaven na objektech, které jsou stavebními bloky celého úložistě. Ať už se jedná o jakýkoliv formát dat nebo soubor, jsou tyto data uloženy ve formě objektů v CEPH clusteru. CEPH pak díky tomu, že využívá objektové úložiště umožňuje dosáhnout platformní a hardwarovou nezávislost. Kromě objektového úložiště CEPH poskytuje bloková zařízení přes RADOS nebo iSCSI a také POSIX kompatibilní souborový systém CephFS. Dále také replikuje jednotlivé objekty napříč clustrem a tím zvyšuje spolehlivost úložiště [3].

3.3.2 VMware vSphere

V naší infrastruktuře využíváme VMware vSphere pro tvorbu a správu virtuálních strojů. VMware je jedním z hlavních softwarů pro využití virtualizace v cloudové oblasti. VMware obsahuje rozsáhlé portfolio produktů. VMware vSphere je právě jedním z nich. Aktuálně je na trhu již přes 15 let a je jedním z nejpoužívanějších platform pro datová centra a cloud. VMware vSphere se skládá z několika částí jako je například ESXi, hypervizor, vCenter Server a další. Kompletní seznam je možné nalézt v knize Mastering VMware VSphere 6.7 [4]. Pro vykonávání každodenních činností obsahujících správu virtualizovaných strojů se využívá VMware vSphere Client. Tento klient je postaven na základu HTML5 a umožňuje provádět téměř kompletní obsluhu. Tento systém je propojen s Active Directory a je tedy možné využít doménový účet. Samotný vSphere má zde také vlastní monitorovací systém, ve kterém je možné sledovat stav jednotlivých instancí, sítě a samotného serveru. V našem případě je nutno monitorovat jednotlivé stroje běžící v prostředí VMware, aby bylo možné identifikovat jakýkoliv výpadek nebo problém. Dále je nutno monitorovat zda VMware běží. Na obrázku 3.3 je možné vidět architekturu VMware vSphere.

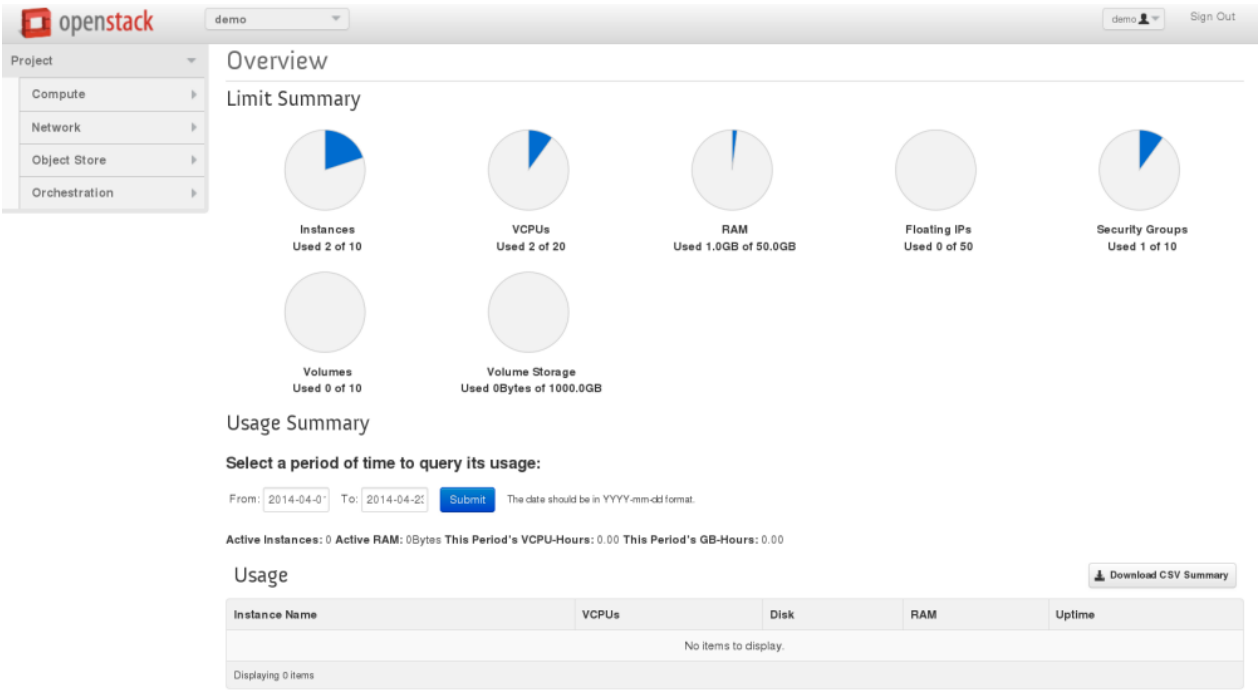


Obrázek 3.3: Schéma VMware vSphere architektury [5]

3.3.3 Openstack

Openstack je soubor softwarových nástrojů, které slouží k provozu a správě cloudové platformy pro veřejné a privátní cloudy. Openstack je vydáván jako open source a je tedy možné přistupovat k jeho zdrojovému kódu a upravit jej. Openstack neustále vyvíjen, díky jeho komunitě, která se skládá z několika tisíců vývojářů po celém světě. Openstack umožňuje uživatelům vytvářet virtuální stroje a spravovat jednotlivé instance. Openstack se skládá z devíti klíčových komponentů, které jsou také distribuovány prostřednictvím každého Openstack systému. Jedná se například o komponent s názvem Nova, který zajišťuje tvorbu a správu virtuálních strojů a veškeré výpočetní úlohy. Openstack vystavuje přehledné webové rozhraní, které zajišťuje propojení všech komponentů a umožňuje na ně vystavovat volání přes HTTP/HTTPS REST API. V tomto rozhraní je možné vytvářet a spravovat veškeré virtuální stroje, pracovat s úložištěm, vytvářet obrazy disků a operačních systémů pro jednotlivé instance a také pracovat s virtuální sítí. Openstack obsahuje jako VMware vlastní monitorovací rozhraní pro monitoring chodu celého systému [6]. Na obrázku 3.4 je vidět prostředí

Openstack.



Obrázek 3.4: Webové rozhraní Openstack [7]

3.4 Třetí úroveň

Ve třetí úrovni již běží samotné virtualizované stroje pro správu infrastruktury. Jednotlivé služby infrastruktury běží na systémech s operačními systémy CentOS a Windows Server. Pro přehled služeb, které využíváme v rámci infrastruktury a které je třeba monitorovat, jsem vybral pár z nich.

3.4.1 pfSense

První a také jednou z hlavních služeb je služba pfSense, která se stará v rámci infrastruktury o PXE boot, přidělování IP adres na základě DHCP protokolu, vytváření jednotlivých DHCP poolů IP adres a poskytování DHCP serveru a firewallu pro jednotlivé VLAN. Dále se také stará o NAT a firewall. pfSense je aktuálně jedním z nejpoužívanějších open source projektů pro správu sítě a firewallu. Jeho využití je možné v nejrůznějších prostředích. Je poměrně lehký na správu a konfiguraci díky webovému rozhraní. pfSense díky jeho komunitě obsahuje velké množství pluginů, které je možné doinstalovat [8]. V našem případě jsou jedny z těchto pluginů Telegraf a Suricata. O Telegrafu se zmiňuji později v práci 5.2.3. Jelikož se jedná o jeden z hlavních prvků infrastruktury je důležité tento systém také monitorovat, aby se předešlo chybám a výpadkům. Jak jsem již zmínil součástí pfSense je také firewall, který je také třeba monitorovat v případě útoků, či nezvyklého chování.

3.4.2 Active Directory

Další ze služeb je Active Directory, která se stará o správu domény infrastruktury, o nastavování skupin, tvorbu uživatelů a kompletní správu uživatelů a správu organizační struktury. Active Directory je služba běžící na Windows Serveru. Jedná se o službu, která se stará o správu informací uživatelů, aplikací a zdrojů a ukládá tyto informace do databáze. Tato databáze umožňuje uložení až dvou bilionů objektů. Uživatelé se na základě informací uložených v Active Directory mohou následně v rámci domény autentizovat a přihlašovat se do různých systémů pomocí jednoho doménového účtu. Dále je možné na základě těchto doménových účtů spravovat přístupy do složek a také omezit přístupy do systémů. Administrátoři mohou spravovat jednotlivé identity z jednoho centrálního úložiště a nedochází k duplikování identit mezi různými systémy [9]. Pro monitorovací systém je tedy nutné, aby podporoval propojení s Active Directory a také aby mohl monitorovat jednotlivé části systému, jako například počet uživatelů, bezpečnost uživatelů v rámci změny hesel a podobně.

3.4.3 VPN

Další z hlavních služeb, které na infrastruktuře běží, je VPN. Díky VPN se uživatelé infrastruktury mohou k infrastruktuře připojit a pracovat na ní a dostat se na její služby. Pro VPN server využíváme Windows Server od Microsoftu, a konkrétně službu Remote Access. Díky Routing and Remote Access klienta je možné nastavovat síťové rozhraní, routy a jednotlivé porty. Díky Remote Access Manegement konzole je možné také vidět aktuálně připojené uživatele, velikost přenesených dat, délku připojení a další informace. Tyto informace je tedy třeba také monitorovat a mít o nich přehled.

3.4.4 HAProxy

Dalším nástrojem, který využíváme, je HAProxy. HAProxy neboli High Availability Proxy je open source nástroj pro TCP a HTTP load balancing. Jeho hlavní funkcí je zlepšovat výkon a spolehlivost serverů díky tomu, že umožňuje distribuovat zatížení mezi servery. HAProxy poskytuje load balancing na síťové a aplikační vrstvě. Nástroj HAProxy je využíván ve velkých prostředích, jako jsou například GitHub nebo Instagram [10]. V našem případě potřebujeme monitorovat vytížení, přenesený počet dat, popřípadě chyby na straně serveru a odpovědí.

3.5 Čtvrtá úroveň

Čtvrtou úrovní infrastruktury jsou již samotné instance používané pro vývoj a testování platformy LEXIS [2], které běží jako virtualizované stroje v prostředích VMware a Openstack. Jedná se o služby, které u nás provozují partneři projektu LEXIS, kteří platformu využívají. Jednotlivé instance této platformy je nutno také monitorovat, abychom zajistili co největší dostupnost konkrét-

ních instancí a minimalizovali jakékoliv problémy a výpadky těchto služeb. Samotná Lexis platforma využívá vlastní monitorovací systém, který ale částečně využívá i monitorovací systém naší infrastruktury. V rámci této práce ale jednotlivé instance služeb LEXIS platformy nebudu popisovat, protože je to mimo kontext této práce.

3.6 Analýza požadavků infrastruktury na monitoring

Díky popisu jednotlivých úrovní je nyní jasné, jaké služby a prvky infrastruktury je třeba monitorovat. Monitorovací systém by měl být schopen zabezpečení a propojení s Active Directory. Další funkcí, kterou by měl monitorovací systém disponovat, je přehledné uživatelské rozhraní, kde bude možné zobrazit a shlukovat různé druhy monitorovaných metrik a služeb. Dále, díky tomu, že infrastruktura obsahuje velké množství serverů, které využívají různé operační systémy a datové zdroje, je nutné, aby monitorovací systém disponoval schopností připojení různých typů a zdrojů dat.

Protože je třeba monitorovat jak hardware, tak software, je důležité, aby monitorovací systém umožnil v jednom prostředí zobrazovat všechna data, ať už se jedná o HW nebo SW a nebylo nutno využívat více systémů pro monitoring. Monitorovací systém by dále měl být schopen vytvářet alerty a odesílat je prostřednictvím komunikačních kanálů v případě jakékoli poruchy nebo problému. Systém by měl být dále snadný na údržbu, konfiguraci a měl by mít možnost zobrazovat data v reálném čase a také se vracet do minulosti, v případě možné potřeby procházet historická data a na základě nich identifikovat problémy, které nastaly nebo mohly nastat.

Jedním z požadavků na monitorovací systém je také to, aby šlo vytvářet vlastní dashboardy na základě uživatele a využívat možnost zobrazení dashboardů konkrétním týmům a eliminovat neoprávněné zobrazení a zasahování do systému. Na základě těchto požadavků bylo nutné provést rešerši monitorovacích nástrojů a vybrat z nich ten nejvhodnější. Rozbor těchto monitorovacích nástrojů je popsán v další kapitole.

Kapitola 4

Rozbor monitorovacích systémů

Při výběru monitorovacího systému pro infrastrukturu je nutné vzít v úvahu několik faktorů, jako je možnost vytváření vlastních dashboardů, možnost úpravy samotného systému, využití aletrů přímo v monitorovacím systému nebo také navázání různých zdrojů dat. Většinou taky ve výběru hraje roli samotná cena a vizuální prostředí systému. Pro potřeby infrastruktury vycházím z předešlé analýzy. Monitorování jednotlivých úrovní infrastruktury vyžadovalo rozsáhlou rešerši různých druhů monitorovacích systémů. Pro přehled jsem vybral několik komerčních a opensource monitorovacích nástrojů, které se používají v současné praxi.

4.1 Kibana

Následující rozbor vychází z knihy Kibana essentials [11] a dokumentace společnosti Elastic, ve které je podrobně popsán nástroj Kibana a jeho funkce [12]. Kibana je open source vizualizační platforma pro zobrazování a analýzu velkého množství dat ve formě grafů, která je postavena nad platformou Elasticsearch a využívá její funkce. Je součástí ELK stacku a je vytvořena a spravována společností Elastic. ELK stack je skládá ze tří hlavních komponent.

1. Logstash

   Logstash je open source nástroj, který umožňuje sběr dat z různých datových zdrojů. Tyto data pak později ukládá do Elasticsearch. Má podporu přes 200 různých pluginů a proto umožňuje uživatelům využít data různého typu nebo zdroje.

2. Elasticsearch

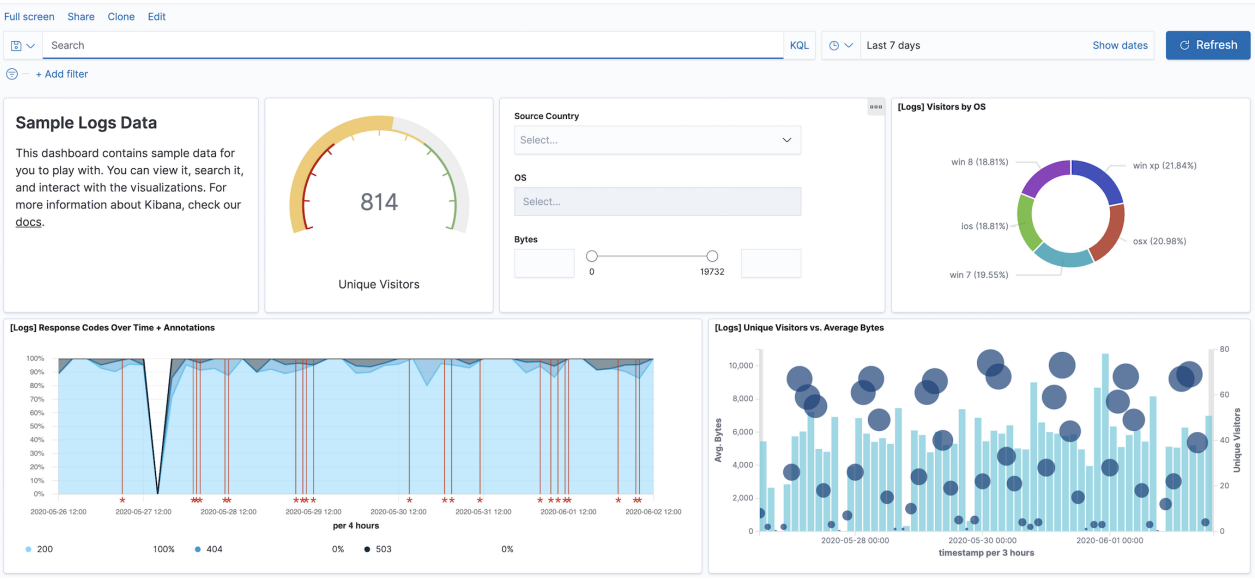
   Elasticsearch je open source vyhledávací server postavený na enginu Apache Lucene, který je napsaný v jazyce Java. Implementuje datovou strukturu zvanou invertovaný index, která umožňuje rychlé fulltextové vyhledávání. Dále poskytuje RESTové rozhraní a podporuje JSON. Podporuje distribuované vyhledávání a jedná se o škálovatelný vyhledávací nástroj umožňu-

jící vysokou flexibilitu při přidávání instancí. Využívá data, která jsou mu poskytnuta přes Logstash, případně přes REST API.

3. Kibana

   Samotná Kibana slouží jako vizualizační nástroj nad Elasticsearch serverem a umožňuje vytváření graficky přívětivých grafů na základě dat, které získává z Elasticsearche. Má velmi dobře zpracované GUI a snadnou manipulaci. Poskytuje různé typy tabulek, histogramů a map. Umožňuje vytváření vlastních dashboardů a jednotlivých panelů. Tyto dashboardy jsou snadno přenositelné, díky jejich reprezentaci pomocí JSON. Umožňuje také zobrazování dat v reálném čase. Pro spuštění vyžaduje webový server a připojení k Elasticsearch pomocí REST API.

Velmi často se právě Kibana používá jako monitorovací systém pro infrastrukturu nebo aplikace. V prostředí infrastruktury je pomocí Kibany možné monitorovat například systémový výkon, sledovat aplikační logy nebo vytvářet aletry pro nastavené thresholdy. Díky podpoře různých pluginů je také možné získávat data z různých zdrojů.



Obrázek 4.1: Ukázka prostředí z nástroje Kibana [12]

- Výhody ELK stacku:
  - Snadná instalace a nastavení
  - Fulltextové vyhledávání
  - Jednotná syntaxe pro dotazy nad zdrojem dat
  - Přehledné uživatelské rozhraní a snadná vizualizace dat

- Nevýhody ELK stacku:
  - Jelikož je Kibana postavená na Elasticsearch, nepodporuje žádný jiný zdroj dat než Elasticsearch.
  - Dostupnost Kibana dashboardů pro veřejnost bez použití různých opensource nebo komerčních balíčků pro autentizaci.
  - Pro vytváření alertů, nutné doinstalovat balíčky jako v případě autentizace.
  - Možná potřeba instalace zpoplatněných balíčků pro rozšíření funkcionality.

4.2 Zabbix

Následující text vychází z knihy [13] a dokumentace pro Zabbix [14]. Zabbix je open source distribuovaný monitorovací systém, který byl vytvořen Alexei Vladishevem a je aktivně vyvíjen a podporován společností Zabbix SIA. Umožňuje monitorovat desetitisíce serverů, virtuálních zařízení a síťových prvků. Zabbix dokáže sbírat data z různých zdrojů jako jsou například aplikace, databáze, virtuální instance a cloudové úložiště.

Další důležitou věcí co nabízí je také možnost vytvoření alertů a odesílání oznámení prostřednictvím emailu. Dále nabízí uživatelsky přívětivé webové rozhraní obsahující různé statistiky a reporty, a ve kterém je možné provádět i změny samotných konfiguračních parametrů. Díky tomu, že je Zabbix distribuován pod licencí GPL verze 2 je jeho zdrojový kód volně dostupný veřejnosti a je tedy možné si jej také přizpůsobit svým potřebám. Samotná konfigurace tohoto systému je dopodrobna popsána ve velmi dobře zpracované a přehledné dokumentaci [14]. Přesto, že je Zabbix open source, nabízí zákazníkům větších společností možnost placené zákaznické podpory. Tato podpora je rozdělena do balíčků podle potřeb zákazníka a je rozdělena na balíčky Silver, Gold, Platinum, Enterprise a Global I. Pro zmínku obsahují například vzdálenou podporu a identifikaci problémů, trénink administrátorů v místě jejich společnosti, měsíční meetingy, předkonfigurované agenty apod.

Architektura monitorovacího systému Zabbix je skládá z několika komponent. Jedná se o Zabbix server, databázové úložiště, webové rozhraní, proxy server a agenty. Jednotlivé komponenty jsou popsány níže.

1. Zabbix server

   Jedná se o hlavní prvek celého monitorovacího systému. Zabbix Serveru jednotliví agenti a proxy server předávají data o dostupnosti a integritě systémů. Server může také sám vzdáleně kontrolovat webové služby, například webové a poštovní servery. Na serveru je uložena veškerá konfigurace v databázi. Dáje je server také součástí systému, který se podílí na vytváření alertů. Funkce serveru je rozdělena na tři komponenty. Jsou to Zabbix Server, webové rozhraní a databáze. Zabbix využívá operační systém Linux.

2. Databázové úložiště

   Pro rychlou odezvu a zpracování dat je pro databázi dobré využít vyšší počet CPU na základě počtu monitorovaných parametrů. Zabbix pro databázi podporuje MySQL s InnoDB enginem, Oracle, PostgreSQL, IBM DB2 a SQlite. Do databáze se ukládají veškeré konfigurace a sbírané data.

3. Webové rozhraní

   Pro snadnou manipulaci a přístup k systému Zabbix na jakékoliv platformě, je vytvořeno přívětivé uživatelské rozhraní. Obvykle samotné rozhraní běží na stejném stroji jako právě Zabbix server, není to ale podmínkou. Ve webovém rozraní je možná konfigurace, tvorba alertů a samotný monitoring instancí, sítě apod.

4. Server proxy

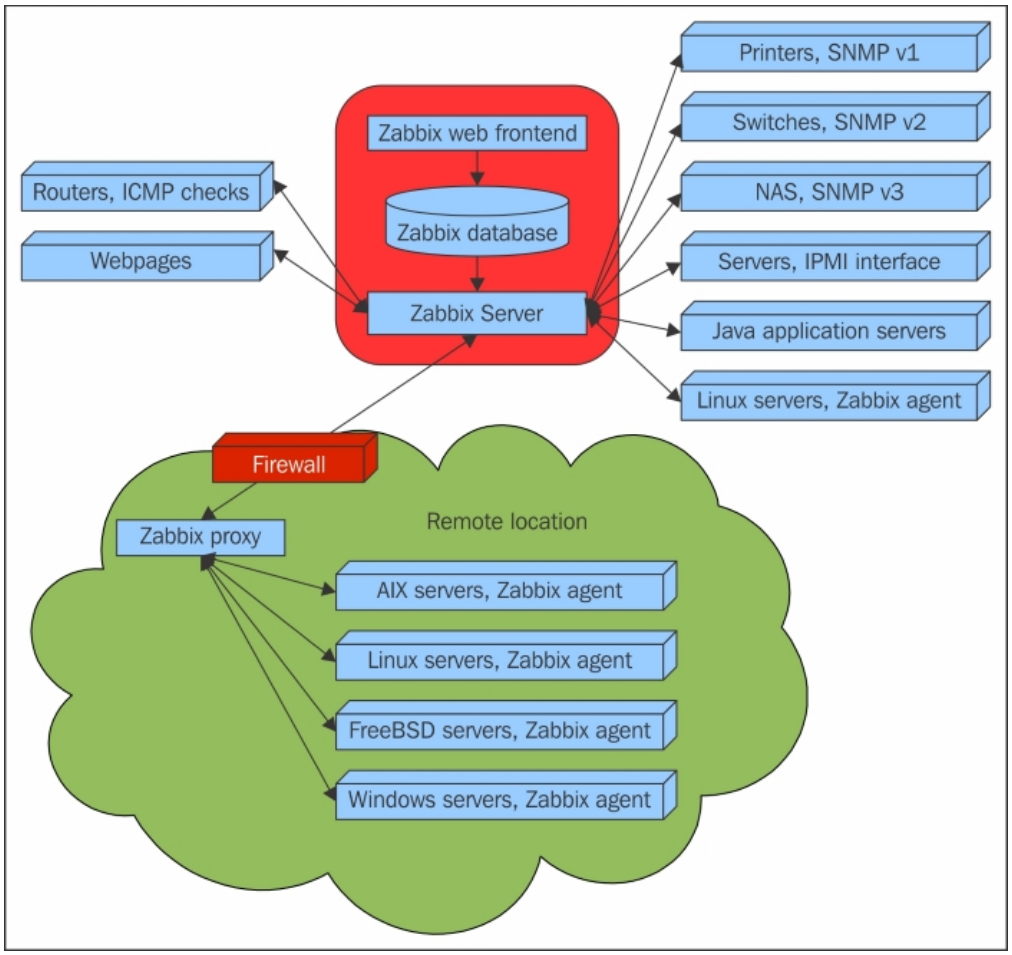
   Zabbix proxy pomáhá ulehčit práci Zabbix serveru. Umožňuje zmírnit jeho zatížení a používá se pro sběr dat. Tato komponenta je volitelná část systému, ale je doporučeno ji využít.

5. Zabbix agent

   Jedná se o agenty, kteří se starají o sběr dat z různých aplikací, serverů apod. a odesílají tyto data Zabbix serveru. Od verze Zabbix 4.4 je možné využít dva typy agentů. Prvním typem je Zabbix agent, který je podporován na většině platforem a je napsaný v jazyce C. Druhý typ je Zabbix agent 2, který je napsaný v jazyce Go a je flexibilní a jednoduše rozšiřitelný pomocí různých pluginů.

- Výhody systému Zabbix:
  - Podpora velkého množství pluginů.
  - Open source a dostupnost kódu
  - Webové rozhraní s možností konfigurace
  - Sběr různých zdrojů dat
- Nevýhody systému Zabbix:

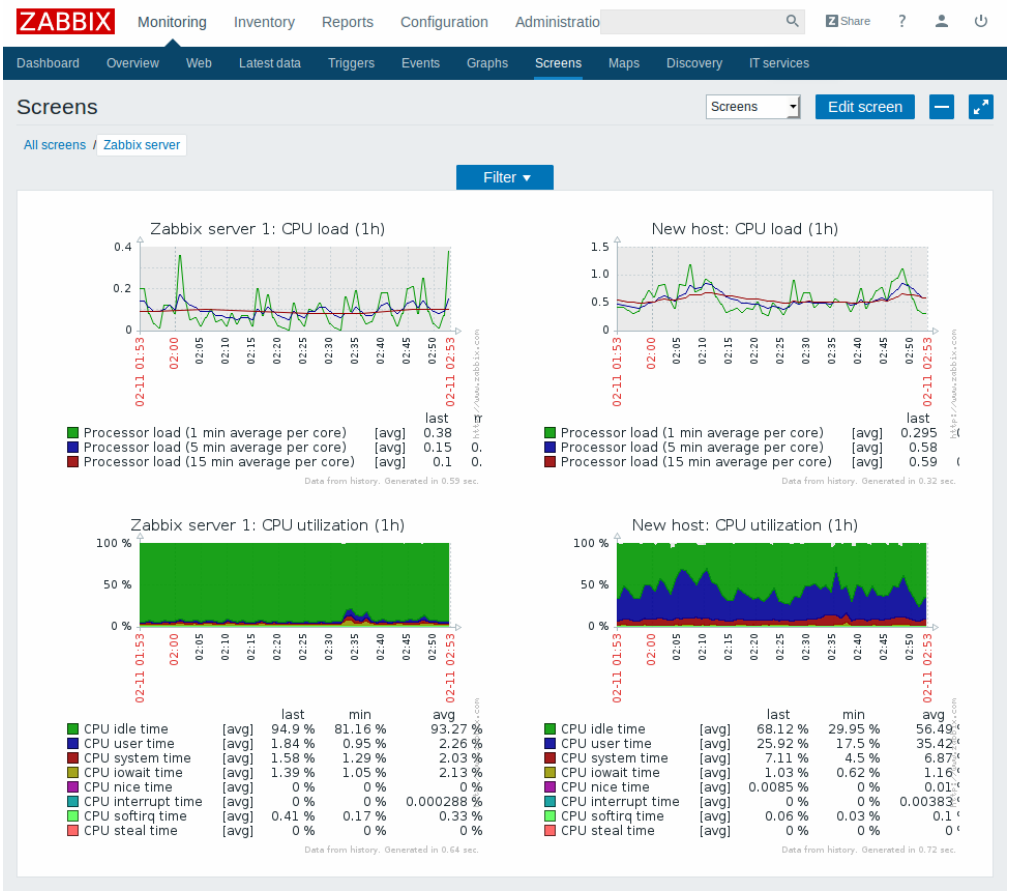
  Uživatelské rozhraní a samotná interpretace grafů není tak pěkná jako například v případě Grafany. To je ale spíše subjektivní pohled a každý by si měl obrázek udělat sám. Samotný Zabbix nabízí právě použití Grafany jako prezenční vrstvu díky lepší vizualizaci grafů. Více informací je možné nalézt v dokumentaci [14].



Obrázek 4.2: Architektura monitorovacího systému Zabbix [13]

4.3 Nagios

Následující text rozboru vychází z knihy Learning Nagios 4 [15] a dokumentace nástroje Nagios [16]. Nagios je monitorovací systém, který podporuje monitorování sítě, aplikací, procesů a umožňuje vytváření a odesílání alertů přes komunikační kanály. Jako například Zabbix umožňuje monitorovat velké množství instancí a jeho limity jsou dány spíše hardwarovým vybavením. Je používán v mnoha velkých společnostech a je rozdělen na open source a komerční řešení. Open source řešení je zdarma se nazývá Nagios Core. Toto řešení je licencováno GNU licencí verze 2. Komerční řešení je nazváno Nagios XI a je v případě potřeby dostupné zdarma na 30 dní. V případě zakoupení stojí standardní edice 1995 dolarů a Enterprise edice 3495 dolarů [17]. Každý balíček nabízí pak jiné služby. Nagios pro svůj běh využívá operační systém Linux. Klíčovými rozdíly mezi Nagios Core a Nagios XI jsou jednoduchost obsluhy a používání samotného systému, technické požadavky na uživatele a zpracování dat pro různé reporty v rámci obchodních potřeb firmy. Nagios Core umožňuje použití velkého množství pluginů a jeho modifikaci. To ale vyžaduje také větší zkušenost administrátorů,



Obrázek 4.3: Webové rozhraní monitorovacího systému Zabbix [14]

čas a podrobné pochopení systému. Nagios XI je narozdíl od Core postaven tak, aby ho zvládali obsluhovat i méně technicky zdatní uživatelé a aby byl přizpůsobený pro danou společnost hned od začátku.

Architektura monitorovacího systému Nagios se skládá z několika komponent. Jedná se o server-agent architekturu. Hlavní z nich je Nagios Core nebo XI. Další je databáze, různé pluginy a nakonec webové rozhraní.

1. Nagios Core, XI

   Hlavním prvkem monitorovacího systému je Nagios Core nebo v případě komerčního řešení Nagios XI. Nagios je nainstalován na zařízení a odesílá signály na agenty, kteří sbírají data na jednotlivých serverech

2. Databázové úložiště

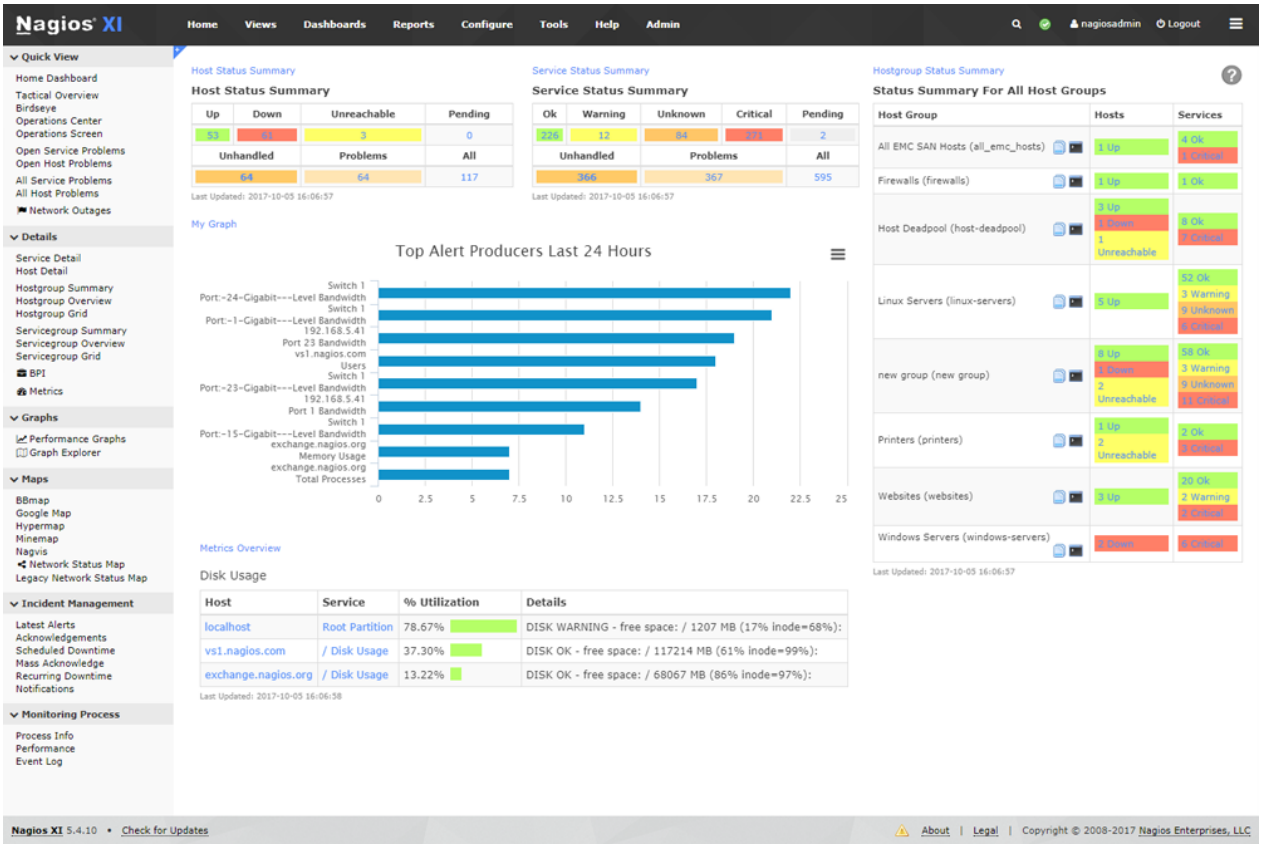
   Pro úložiště dat využívá Nagios databáze typu MySQL nebo MariaDB. V této databázi jsou uložena všechna data, která jsou sbírána agenty.

3. Pluginy

   Pluginy sbírají data z různých instancí a zařízení. Je možné je konfigurovat a upravovat uživatelem. Jelikož je řešení Nagios na trhu již dlouho obsahuje velké množství těchto pluginů pro různé rozšíření systému a sběr různého typu dat

4. Webové rozhraní

   Ve webovém rozhraní je možné vytvářet a zobrazovat grafy na základě různých typů dat. V případě Nagios XI je možné ve webovém rozhraní provádět také konfiguraci a vytvářet dashboardy pro konkrétní uživatele. Dále je také možné vytvářet podrobné reporty.



Obrázek 4.4: Webové rozhraní monitorovacího systému Nagios XI [16]

- Výhody systému Nagios:
  - Velké množství pluginů.
  - Možnost vytváření vlastních dashboardů pro konkrétní uživatele.
  - Jednoduchost použití v případě Nagios XI.
  - V základu podpora alertů.

- Nevýhody systému Nagios:
  - Cena.
  - Pro rozšířené prvky v systému nutné vlastnit komerční verzi.
  - Složitost samotné konfigurace systému.
  - Z mého pohledu příliš složité a matoucí webové rozhraní.

4.4 Icinga 2

Následující text vychází z článku Tools and strategies to monitor the ATLAS online computingfarm [18] a dokumentace pro projekt Icinga [19]. Icinga2 je open source monitorovací systém, který je snadno rozšiřitelný pomocí různých pluginů a nabízí široké možnosti monitorování infastruktury. Umožňuje funkce, jako jsou monitoring pomocí SNMP, vytváření alertů, monitoring stavu instancí, sledování dostupnosti procesů a nabízí přehledné webové rozhraní. Projekt Icinga byl vytvořen jako fork systému Nagios kvůli snaze vyvinout systém, který by byl více orientován na komunitu a adaptoval se na dynamicky rostoucí požadavky na nynějším trhu. Dalším podnětem pro vytvoření tohoto systému bylo vylepšení webového rozhraní a usnadnění konfigurace samotného systému.

Díky tomu, že je projekt Icinga vytvořen na základě projektu Nagios, je zachovávaná kompatibilita s jeho dosavadními pluginy. Projekt Icinga je vydáván pod licencí GNU GPL verze 2. Je důležité zmínit, že v základní instalaci systému Icinga není hned možné monitorovat všechny různé služby, ale je nutné doinstalovat potřebné pluginy, kterých je ale velké množství pro různé zdroje dat. Stejně jako například systém Zabbix má i Icinga možnost dokoupení uživatelské podpory, která je rozdělena do několika balíčků na základě uživatelských potřeb. Pro instalaci projektu Icinga je předpokládán operační systém Linux. Samotná instalace a konfigurace je podrobně popsána v dokumentaci na stránkách projektu.

Jako i ostatní monitorovací systémy je Icinga rozdělena na několik komponent, kterými jsou:

1. Icinga2 Server

   Jedná se o základní prvek samotného systému. Obsahuje veškerou konfiguraci a vyžaduje pro instalaci některou z distribucí operačního systému Linux.

2. Icinga2 DB

   Databáze se stará o uložení všech konfigurací a informací o datech. Je možné využít MySQL nebo PostgreSQL. Aktuálně v době psaní této práce je možné pro testovací účely použít i právě vyvíjený Icinga DB backend. Protože se ale jedná o fázi testování, není doporučován pro nasazení na produkční prostředí.

3. Icinga2 API

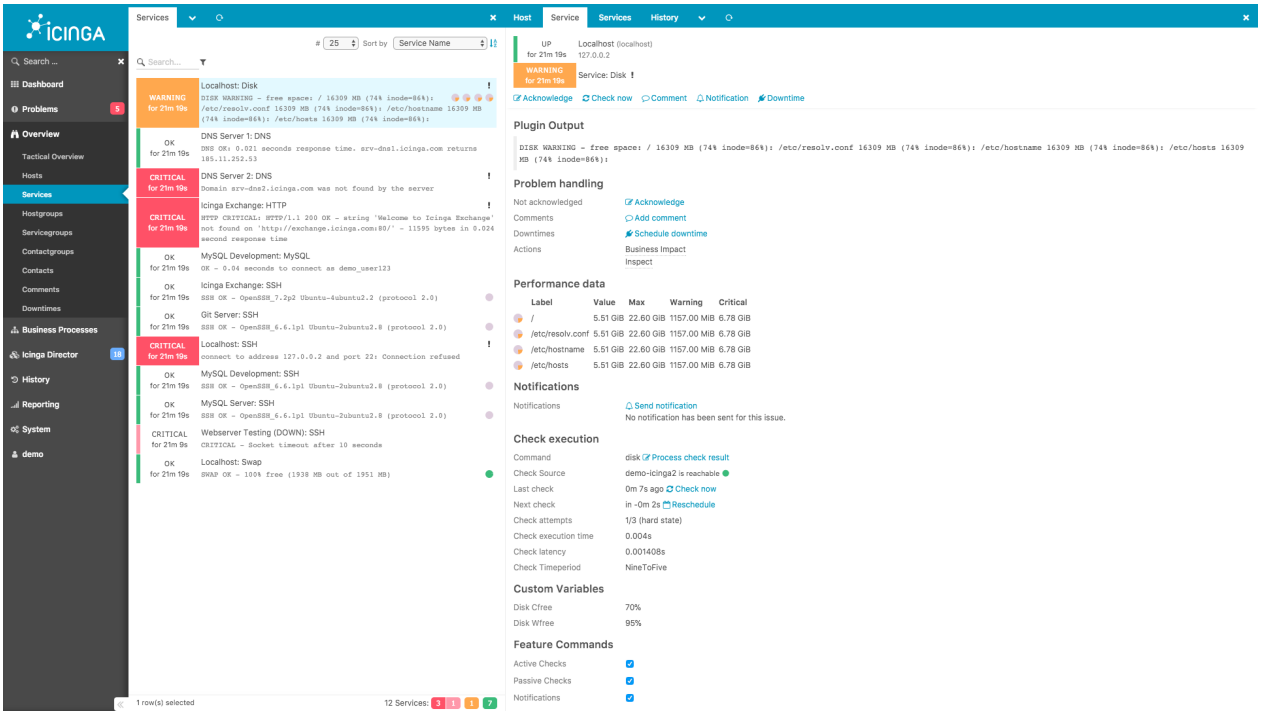
Icinga2 nabízí využití API, které umožňuje propojení s jinými systémy. Dále je možné provádět konfigurace, spouštět skripty a vytvářet či upravovat objekty přes HTTP requesty.

4. Icinga Web 2

   Webové rozhraní, které je přehledně uspořádané a využívá Icinga2 server. Umožňuje seskupovat a kombinovat různé prvky a vytvářet z nich vlastní dashboardy. Umožňuje nastavení alertů a jejich odesílání přes komunikační kanály.

5. Icinga moduly

   Pomocí modulů je možné základní systém Icinga rozšířit o další prvky na základě potřeb uživatele. Jedná se například o modul pro vSphere, který umožňuje kompletní monitoring VMware instancí, úložiště a systému. Dále například modul pro Windows, který sbírá informace o systému Windows z jeho instancí. Těchto modulů je velké množství a díky rozšířenosti v komunitě jich neustále přibývá.



Obrázek 4.5: Webové rozhraní monitorovacího systému Icinga2 [19]

- Výhody systému Icinga2:
  - Real-time monitoring a zobrazování podrobných informací ve webovém rozhraní.
  - Velké množství modulů, které umožňují rozšíření systému a monitorování nejrůznějších zdrojů.

  - Graficky dobře organizované a přehledné webové rozhraní.
  - Možnost tvorby alertů.
  - Možnost využití API a propojení s dalšími systémy.

- Nevýhody systému Icinga2:
  - Složitost počátečního nastavení

4.5 Prometheus

Následující text vychází z knihy Prometheus: Up & Running: Infrastructure and Application Performance Monitoring [20] a dokumentace monitorovacího systému Prometheus [21]. Prometheus je open source monitorovací systém, který byl vyvinut v roce 2012 ve společnosti SoundCloud. Prometheus je primárně napsán v jazyce Go a je licencován pod Apache 2.0 licencí. Projekt Prometheus od svého počátku získal velké množství organizací, které jej využívají jako primární nástroj pro monitoring infrastruktury. Projekt má velmi rozsáhlou komunitu a je aktivně vyvíjen a vylepšován. Aktuálně systém Prometheus používají v produkci desetitisíce společností. V roce 2016 se projekt Prometheus stal druhým členem organizace Cloud Native Computing Foundation (CNCF).

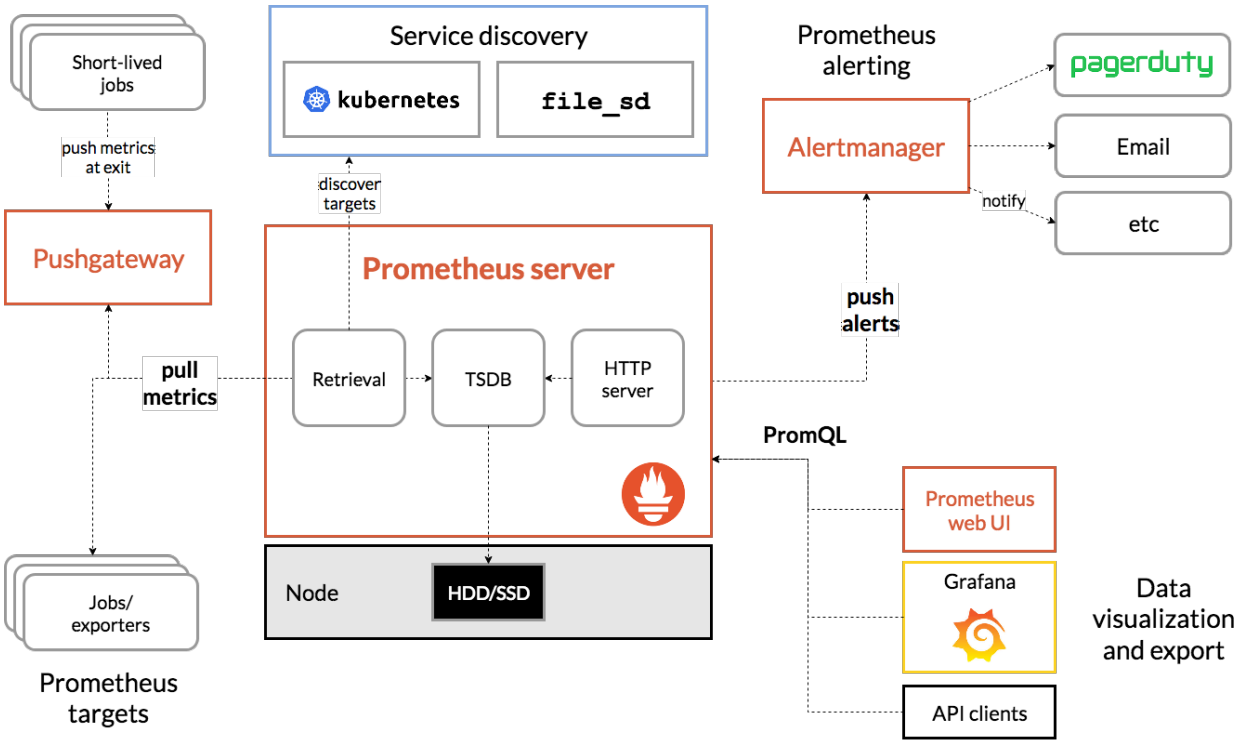
Pro aplikaci vlastního kódu nabízí různé klientské knihovny ve většině jazycích jako například Java, C#, Python, Node.js a Rust. Pro integraci jiných systémů než těch, které podporují formát Promethea, existují různé typy exporterů. Prometheus využívá pro dotazování jazyk PromQL. Díky tomu, že datový model systému identifikuje každou time series pomocí nesetříděného páru klíče a hodnoty, kterou nazývá "label", a ne jen na základě jména, je možné vyhledávat data například na základě služby, procesu, datacentra nebo jakéhokoli jiného nadefinovaného štítku. Díky štítkům je jednodušší tvorba alertů a v jednom alertu je možné monitorovat více strojů nebo procesů, na rozdíl od ostatních systémů, kde je třeba vytvořit pro každou instanci alert zvlášť.

Architektura systému Prometheus se skládá z několika komponentů. Jedná se o Prometheus server, pushgateway, exportery, klientské knihovny, alertmanager, a webového UI. Architektura je blíže zobrazena na obrázku 4.6.

1. Prometheus server

   Prometheus server je hlavním prvkem celého systému, obsahuje veškerou konfiguraci a stará se o ukládání sbíraných time series dat. Prometheus ukládá data lokálně ve vlastní databázi, což také usnadňuje provoz samotného systému. V průběhu let právě úložiště prošlo několika úpravami a nyní je možné pomocí jediného serveru monitorovat tisíce instancí. Pro rychlost zpracování je doporučen SSD disk. Pro konfiguraci se využívá jazyk YAML. Samotná konfigurace je poměrně jednoduchá a je popsána v dokumentaci projektu.

2. Pushgateway



Obrázek 4.6: Architektura monitorovacího systému Prometheus [21]

Pushgateway slouží jako prostředník mezi úlohami, a službami, které nemají dlouhou životnost. Umožňuje jim odesílat data právě na tuto gateway a ta následně data vystaví serveru.

3. Klientské knihovny

   Pro to, aby aplikace mohly poskytovat data pro Prometheus, je nutné implementovat těmto aplikacím klientské knihovny. Tyto knihovny jsou dostupné v různých programovacích jazycích. Projekt Prometheus oficiálně podporuje jazyky Go, Python, Java/JVM a Ruby. Dále jsou zde ale také k dispozici neoficiální klientské knihovny.

4. Exportery

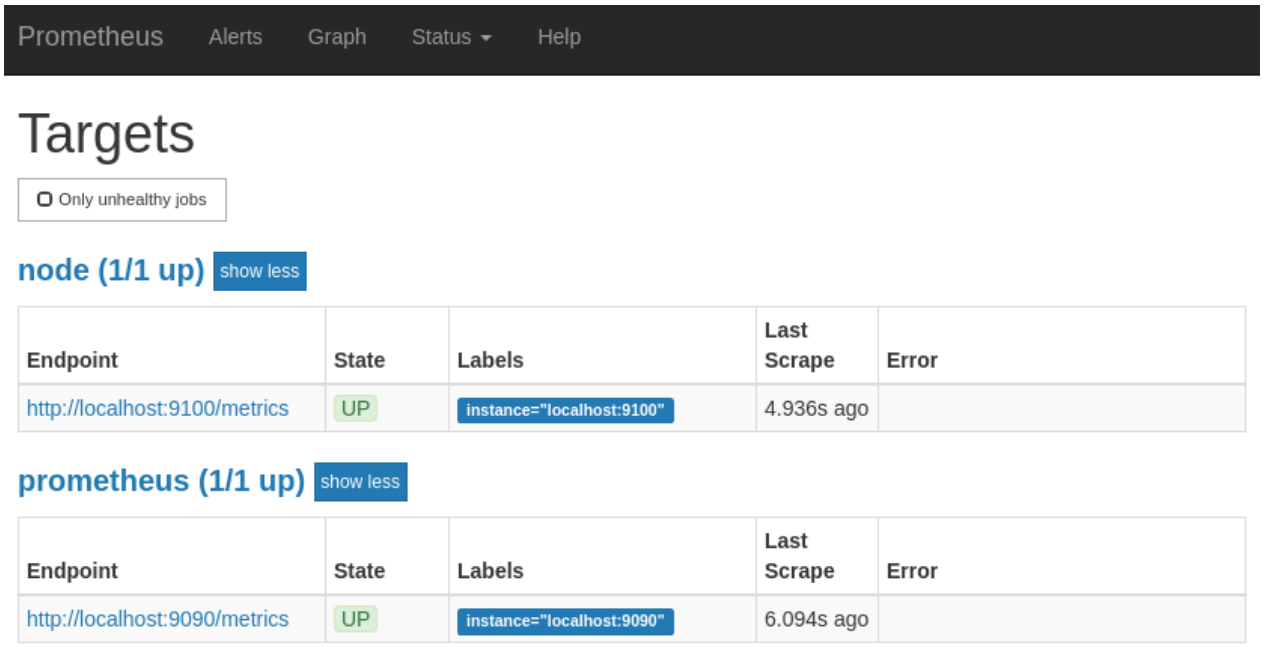
   Exportery jsou něco jako agenti v jiných monitorovacích systémech, jako například v systému Zabbix viz 4.2. Tyto exportery je možné nasadit na instanci nebo například před aplikaci, ze které chceme sbírat data. Exporter obdrží žádost o data od serveru, tyto data vyčte z aplikace, transformuje je do správného formátu a vrátí v odpovědi zpátky serveru. Díky tomu, že projekt Prometheus má obrovskou komunitu, je většina exporterů již vytvořena a je vhodná pro implementaci do vlastního prostředí.

5. Alertmanager

Alertmanager slouží pro odesílání oznámení o alertech prostřednictvím komunikačních kanálů. Alertmanager obdržuje alerty ze serveru a ty následně přeměňuje v notifikace. Je možné využít například email nebo chatovací aplikace.

6. Webové rozhraní

   Prometheus sám o sobě poskytuje webové rozhraní, které umožňuje vytvářet dotazy pomocí jazyka PromQL a zobrazovat data. Slouží také k procházení těchto dat a také například zobrazení monitorovaných instancí. V základu ale toto rozhraní není určeno pro kompletní vizualizaci. Z tohoto důvodu se systém Prometheus nejčastěji pojí s vizuálním prostředím systému Grafana. Samotný Prometheus doporučuje využití systému Grafana a je tímto projektem také oficiálně podporován. Na obrázku 4.7 je vidět webové rozhraní systému Prometheus. Projekt Grafana je popsán dále v práci viz 4.6.



Obrázek 4.7: Webové rozhraní monitorovacího systému Prometheus [21]

- Výhody systému Prometheus:
  - Rozsáhlá komunita a podpora.
  - Open source systém s rozsáhlými možnostmi rozšíření.
  - Snadná konfigurace.
  - Snadná integrace s ostatními systémy a programovacími jazyky.
  - Využití štítků pro interpretaci a vyhledávání v datech.

  - Velké množství exporterů pro nejrůznější systémy, aplikace a zdroje dat.

- Nevýhody systému Prometheus:

  V základu neobsahuje uživatelské rozhraní pro kompletní vizualizaci a tvorbu dashboardů.

4.6 Grafana

Ve spojení s monitorovacím systémem Prometheus je velmi často nasazován vizualizační nástroj Grafana a proto ji považuju za nutné zmínit. Grafana je open source vizualizační nástroj, který umožňuje analyzovat a monitorovat time-series data viz obrázek 4.8. Nástroj je napsán v jazyce Go. Její přednostní je možnost připojení několika různých zdrojů dat viz obrázek 4.9 a její GUI. Umožňuje také nastavení alertů přes různé komunikační kanály na základě analyzovaných dat a vytvořených thresholdů. Grafana také umožňuje vizualizaci různých chyb, aplikačních logů a nejrůznějších typů metrik, ovšem při správné konfiguraci a využití různých zdrojů dat. Dalšími funkcemi jsou například vlastní tvorba dashbordů, možnost rozdělení dashboardů na organizace a týmy a nebo také propojení s Active Directory. Právě díky rozšířeným možnostem použití někteří vydavatelé monitorovacích systémů podporují propojení s Grafanou pomocí různých pluginů nebo API. Jedním z nich je například Icinga nebo Zabbix.

Grafana je poměrně snadná na konfiguraci a stránky dokumentace jsou výborně zpracované. Podporuje velké množství systémů, od nejrůznějších distribucí Linuxu, přes Windows, až po macOS. Právě Grafana je oblíbený vizualizační nástroj ve firmách díky její stále rostoucí komunitě, aktualizacím, rozšiřováním funkcí a podpoře. Společnost pořádá také různé druhy konferencí, kdy je možné se dozvědět o nejnovějších funkcionalitách a také klást dotazy expertům z oblasti vizualizace a analýzy dat [22].

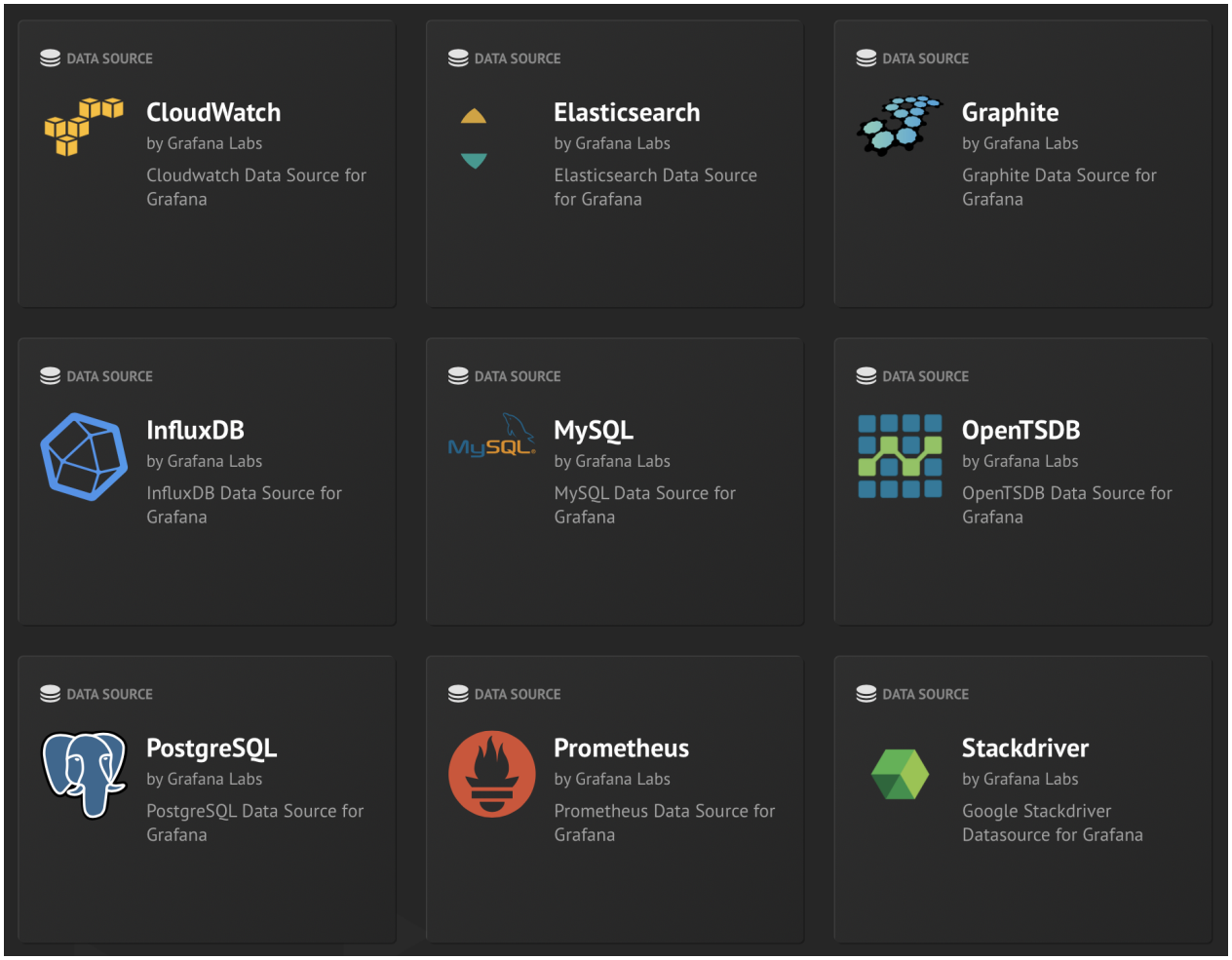
- Hlavní funkce a výhody

  - Možnost zpracování různých zdrojů dat. Některé nástroje jsou vázány pouze na konkrétní databáze popřípadě systémy. Grafana umožnuje připojení přes 30 různých databází z nichž jsou například MySQL, PostgreSQL a InfluxDB. Dále umožňuje připojit různé monitorovací systémy pomocí pluginů a API. Všechny tyto zdroje dat je pak možné slučovat do dashboardů a vytvářet komplexní interaktivní dashboardy.
  - Open source nástroj, který obsahuje velké množství pluginů a přednastavených dashboardů, které jsou stále vyvíjeny, díky rostoucí komunitě.
  - Webové GUI Grafany patří mezi jedny z nejhezčích pro vizualizaci. Umožňuje zobrazit v grafech osy, popisky dat a interaktivně s grafy pracovat v reálném čase.
  - Možnost vytváření alertů přímo v prostředí a jejich navázání na velké množství komunikačních kanálů.

Obrázek 4.8: Ukázka vizualizace v prostředí Grafana [23]

  - Rychlost zpracování dat a autentizace uživatelů s možností vytváření dashbordů na základě organizací

- Nevýhody systému Grafana:
  - Pro uživatele, který požaduje
  - Uživatel se v případě tvorby vlastních dashboardů a panelů musí orientovat v jazycích pro dotazování a naučit se používat rozdílné syntaxe na základě použitého datového zdroje.



Obrázek 4.9: Příklad datových zdrojů v prostředí Grafana [23]

Kapitola 5

Rozbor využitých technologií a implementace

V této kapitole jsou popsány jednotlivé technologie, které byly použity v rámci tvorby monitoringu infrastruktury na základě požadavků, které byly zmíněny v sekci 3.6. Díky rešerši v oblasti monitorovacích systému 4 bylo možné vybrat konkrétní monitorovací systém, který bude splňovat požadavky pro monitoring komplexní infrastruktury a všech jejich úrovní. V našem případě se nejlépe pro potřeby infrastruktury hodí monitorovací systém Prometheus ve spojení s nástrojem Grafana, který je popsán v sekci 4.5.

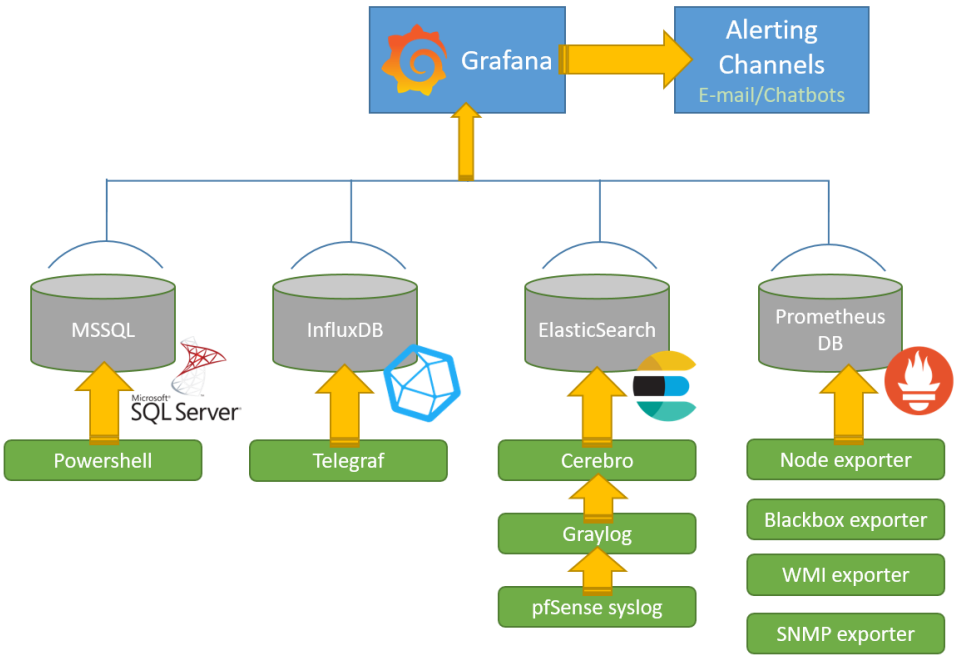
Systém Prometheus ve spojení s nástrojem Grafana byl vybrán z důvodu možnosti připojení velkého množství zdrojů dat, kterými infrastruktura disponuje, a je tedy možné sloučit nejrůznější zdroje do jednoho systému. Dále díky velkému množství pluginů systému Prometheus a široké komunitě je možné monitorovat nejrůznější části a úrovně infrastruktury, které v našem případě potřebujeme. Díky systému Grafana je možné vytvářet alerty pro všechny datové zdroje infrastruktury z jednoho systému a mít tak o všem přehled v rámci konkrétního dashboardu.

Pro potřeby a nasazení monitorovacího systému a všech jeho částí pro monitorování všech úrovní infrastruktury, je nutné mít přehled v široké oblasti technologií, ať už se jedná o skriptovací jazyky, pokročilé znalosti administrace UNIXových serverů, správu databází a také o obsluhu různých pluginů, které v našem případě je nutné použít. Díky tomu, že Prometheus je jako takový komplexní monitorovací systém a má velké množství pluginů je třeba se naučit pracovat i s těmito pluginy pro využití v monitoringu. V první sekci této kapitoly je popsáno nasazení monitorovacího systému Prometheus a nástroje Grafana.

5.1 Nasazení monitorovacího systému

Pro nasazení kompletního monitorovacího systému bylo třeba nasadit vizualizační nástroj Grafana s různými typy datových zdrojů, které popisuji v sekci 5.2. Hlavním prvkem, do kterého jsem

přidal všechny datové zdroje a který slouží pro vizualizaci všech těchto sbíraných dat je Grafana. Protože Grafana se spojuje se systémem Prometheus, jsou oba tyto systémy nasazeny na jednom virtuálním stroji, který má přiřazeny 2 CPU, 6 GB RAM, a 1TB disk. Oba nástroje nasazuji na 64 bitovém operačním systému CentOS7. Veškerá konfigurace a implementace pro systém Grafana a Prometheus v této práci je určena pro systém CentOS7. Ostatní datové zdroje jsou popsány dále v práci. Výsledný monitorovací systém je vidět na schématu níže.



Obrázek 5.1: Schéma monitorovacího systému s exportery a datovými zdroji

5.1.1 Nasazení vizualizačního nástroje Grafana

Pro nasazení vizualizačního nástroje Grafana jsem využil oficiální dokumentaci, kterou je možné nalézt na stránkách GrafanaLabs viz [24]. Prvním krokem pro nainstalování nástroje je vytvoření repozitáře pro YUM. Díky tomu je možné nástroj následně aktualizovat pomocí aktualizování balíčku. Po vytvoření tohoto souboru je již možné nástroj Grafana nainstalovat. Po nainstalování balíčku provádím spuštění procesu, kontrolu, zda naběhl v pořádku, a nastavení automatického spuštění procesu po startu systému. Po nainstalování je možné dostat se do webového rozhraní nástroje přes IP adresu stroje a port 3000. V případě, že by se webové rozhraní nenačetlo, doporučuji ověřit, zda je port 3000 otevřený. V základu je pro přihlášení uživatelské jméno a heslo admin. Po přihlášení systému vyzve uživatele ke změně hesla.

Po nainstalování a prvním spuštění jsem provedl konfiguraci nástroje. Podrobnou konfiguraci je možné nalézt na konci práce viz B. Díky tomu, že v infrastruktuře využíváme Active Directory

a Grafana umožňuje propojení propojení pomocí LDAP protokolu, rozhodl jsem se Grafanu s AD propojit. Toto propojení umožní přístup, editaci a administraci nástroje Grafana pro uživatelské skupiny a konkrétní uživatele na základě definovaných pravidel. Grafana je defaultně nainstalována v cestě:

```
/etc/grafana/grafana.ini
```

Tento soubor je nutné upravit a přidat do něj potřebnou konfiguraci. Podrobný popis konfigurace je možné nalézt na stránkách dokumentace [24].

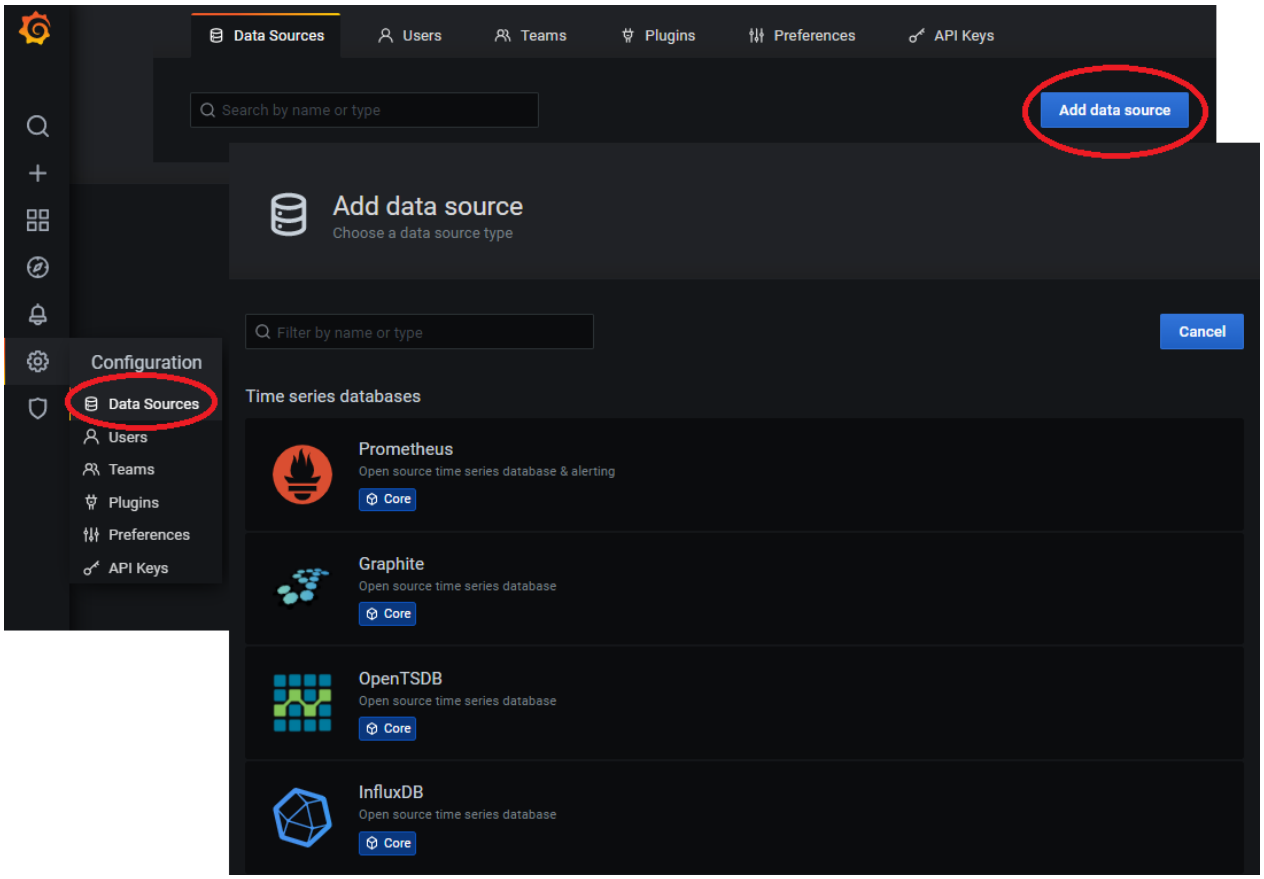
Dále je třeba upravit soubor ldap.toml, který obsahuje informace o doméně, serveru na kterém běží AD, informace o skupinách, cestu k organizační jednotce a další informace. Pro účely propojení s AD je třeba zmínit, že je nutné mít vytvořeného v AD uživatele, který bude sloužit pro čtení organizační struktury a bude vystavovat pro Grafanu informace potřebné k identifikaci uživatelů a k jejich přihlášení. Další velmi důležitou částí propojení je také vytvoření uživatelských skupin v AD, které budou reprezentovat jednotlivé oprávnění pro uživatele Grafany. Grafana má role jako Admin, Editor a Viewer. Tyto role označují možnosti uživatele pracovat se systémem. Díky tomuto namapování na skupiny v AD je možné následně kontrolovat uživatelská oprávnění na základě skupin. Příklad jedné takové skupiny ze souboru ldap.toml je vidět níže.

```
[[servers.group_mappings]]
group_dn = "cn=grafana-administrator,ou=projekt,dc=corp,dc=local"
org_role = "Admin"
grafana_admin = true
```

Po správném nastavení a konfiguraci je třeba proces restartovat. Následně by již mělo být možné využít ověření vůči AD a přihlášení na základě doménového účtu. Pro administraci Grafany je nyní třeba být ve skupině, která má přiřazenou roli Admin. Po přihlášení je možné vidět uživatelské prostředí Grafany a připojit první datový zdroj, kterým je monitorovací systém Prometheus. Protože datových zdrojů, které do Grafany připojuji, je více, rozhodl jsem se toto rozdělit do více podsekcí a v každé z nich popsat propojení s určitým nástrojem a popsat, proč konkrétní datový zdroj používám. Propojení v nástroji Grafana se provádí pomocí volby Data Sources v záložce Configurations. Následně se zobrazí menu pro výběr datového zdroje, který chceme využít. Příklad lze vidět na obrázku 5.2.

5.2 Datové zdroje a jejich implementace

V této sekci popisuji proces, jak jsem řešil monitoring všech úrovní infrastruktury. Tento proces se skládal z několika kroků a byly využity různé technologie. Je zde možné najít popis tvorby dashboardů pro konkrétní datové zdroje a práce s nimi, popis monitorovaných metrik a jejich získávání z datových zdrojů a procesů. Dále je zde popsána tvorba alertů v prostředí Grafana. Je zde také



Obrázek 5.2: Přidání datového zdroje v prostředí Grafana

popsáno propojení jednotlivých datových zdrojů s nástrojem Grafana. Tyto jednotlivé datové zdroje a použité technologie pro získávání potřebných dat na základě požadavků monitoringu všech úrovní infrastruktury jsou rozepsány do dalších podsekcí. Příklad vizualizace vytvořeného dashboardu na základě různých datových zdrojů je vidět níže na obrázku 5.3.

5.2.1 Prometheus

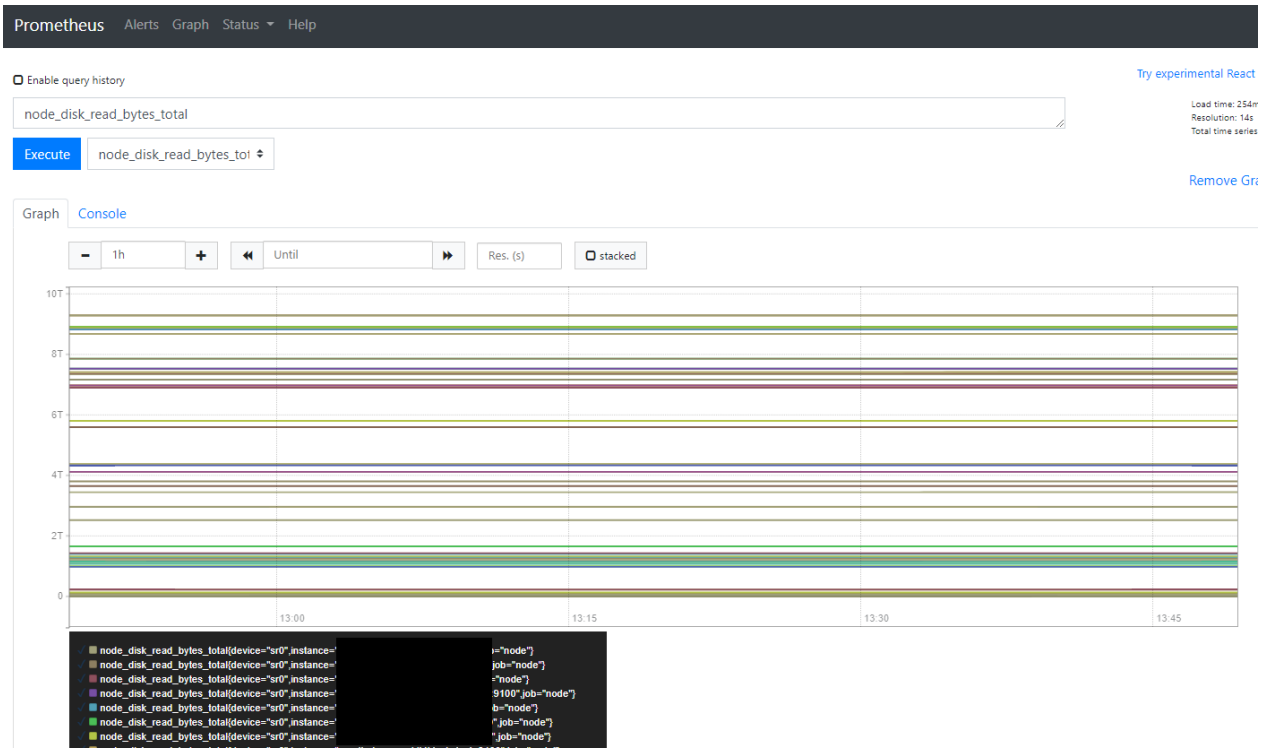
Prvním datovým zdrojem, který jsem do Grafany připojil, je Prometheus. Jedná se o jeden z hlavních prvků monitorovacího systému. Na základě rozboru požadavků na monitoring infrastruktury je třeba monitorovat jednotlivé instance, ať už se jedná o Windows, nebo distribuce UNIXových systémů. Také je třeba monitorovat jednotlivé zařízení, na kterých běží virtualizační platformy a také úložiště. Z toho důvodu jsem implementoval systém Prometheus s několika pluginy, které jsou dále popsány. Prometheus jsem nasadil pomocí automatizačního nástroje Ansible, který umožňuje snadné a bezpečné nasazení softwaru na různý počet instancí. Funguje na bázi SSH protokolu a umožňuje použití sudo. Jeho výhodou je také to, že nepřepisuje soubory, které nevyžadují změnu a nasazení nebo konfiguraci je možné spustit například pro konkrétní skupinu instancí. Tímto způso-

Obrázek 5.3: Komplexní dashboard vytvořený z různých datových zdrojů.

bem není třeba konfigurovat na každé instanci systémy zvlášť, ale je možné využít centralizovaného nasazení na konkrétní instance. Více o systému Ansible je možné najít na stránkách dokumentace viz [25].

Prometheus nasazuji pomocí scriptu pro Ansible ve formátu YAML, ve kterém určuji, na kterou instanci se má systém nasadit a jaké konfigurace se mají pro systém provést. Ve scriptu také nasazuji jednotlivé agenty, kteří jsou třeba ke sběru dat z konkrétních strojů a instancí. Adresy daných serverů uchovávám v centralizovaném souboru na základě skupin. Tímto je možné určit, na které konkrétní instance chci tyto agenty pro Promethea nasadit. V konfiguraci pak určuji, jak často se mají data z instancí sbírat nebo také jak dlouho se mají data uchovávat. Po konfiguraci a spuštění systému Prometheus je možné k němu standardně přistupovat na portu 9090. V tomto rozhraní je možné vykonávat dotazy na zjištění informací o sbíraných metrikách a zobrazovat je v grafech. Jak jsem již v rozboru technologií zmínil, toto uživatelské rozhraní není úplně přívětivé a proto se pojí Prometheus právě se systémem Grafana. V rozhraní si je možné také zobrazit konfiguraci, jednotlivé instance, které jsou monitorovány, a jaké agenty prometheus aktuálně využívá na jakých instancích. Příklad webového rozhraní se zobrazením grafu je možné vidět níže na obrázku 5.4.

Pro monitorování jednotlivých instancí využívám pro systém Prometheus aktuálně tři exportery.



Obrázek 5.4: Ukázka grafu z webového rozhraní monitorovacího systému Prometheus

Těchto exporterů je velké množství a podrobný popis přesahuje rozsah této práce. Pro přehled jsem popsal některé z nich níže.

- Blackbox exporter

  Blackbox exporter umožnuje zjišťování stavu na koncových bodech pomocí protokolů jako jsou například HTTPS, DNS, TCP nebo ICMP. Díky tomuto exporteru monitoruju zda jsou všechny instance v provozu, zda běží například webové rozhraní pro správu systému VMware a nebo také odezvy na DNS.

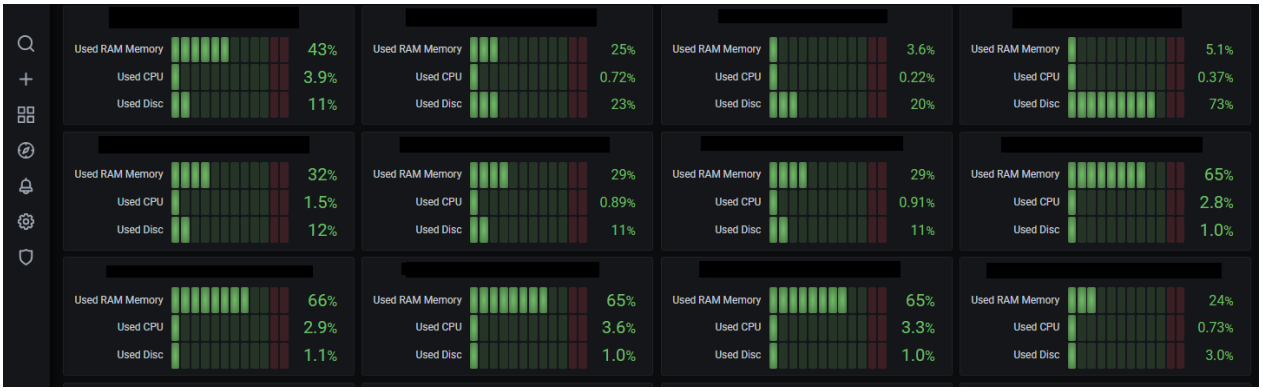
- Node exporter

  Node exporter slouží ke sběru dat a zjišťování informací na UNIXových systémech. Umožňuje sbírat data například o využití CPU, souborovém systému, využití sítě a podobně. Díky node exporteru monitoruji vytížení a chování jednotlivých instancí a využívám možnosti si zobrazit kompletní informace o dané instanci.

- WMI exporter

  WMI exporter je stejně jako node exporter určený ke sběru dat o vytížení, stavu sítě a systémových informací o instancích. Tento exporter je ale určený pro systém Windows. Díky tomu můžu monitorovat chování a stav všech instancí se systémem Windows v rámci infrastruktury.

Po nasazení a úspěšném sběru metrik je možné připojit systém Prometheus jako datový zdroj do Grafany. Díky tomu, že běží na stejném stroji jako Grafana, je možné pro připojení využít localhost adresu. Po připojení je možné vytvářet dotazy a získávat tak data o instancích, které Prometheus monitoruje. Dále je možné vyvářet na základě těchto dat jednotlivé dashboardy. Pro příklad jsem si vybral dva dashboardy pro monitorování vytížení instancí viz 5.5 a stav CEPH clusteru viz 5.6, které je možné vidět níže.



Obrázek 5.5: Ukázka dashboardu pro přehled vytížení instancí

5.2.2 MSSQL

Dalším datovým zdrojem, který je nutné do Grafany připojit je databáze MSSQL. Na základě popisu úrovní infrastruktury a jejich požadavků na monitoring, bylo nutné monitorovací systém rozšířit o další prvky. Prometheus jako takový nenabízí možnost monitoringu služby jako je například VPN Remote Access proces od Microsoftu. Z tohoto důvodu jsem musel přikročit k jinému řešení. Díky tomu, že Grafana umožňuje připojení několika datových zdrojů, rozhodl jsem se pro monitorování služeb VPN využít jako datový zdroj MSSQL databázi.

Pro tvorbu databáze jsem využil Microsoft SQL Server 2019 express, který běží na virtuální instanci se systémem Windows Server 2019. Na serveru jsem si vytvořil databázi a v ní 2 tabulky. Pro VPN potřebuji monitorovat metriky jako jsou například konkrétní přihlášení uživatelé, počet přenesených bytů, login uživatele v doméně a také délku přihlášení. Z tohoto důvodu jedna z vytvořených tabulek obsahuje právě tyto sloupce pro konkrétní metriky. Podrobnou konfiguraci a tvorbu tabulek je možné nalézt na konci práce viz A.

Další tabulkou, kterou jsem si vytvořil, je tabulka pro zaznamenávání informací a stavu o konkrétním procesu. V mém případě se jedná o proces RemoteAccess a RasMan, které jsou hlavními procesy pro chod VPN služby. Z tohoto důvodu je třeba tyto procesy také monitorovat. Protože pro VPN server využíváme v infrastruktuře více instancí, je nutné monitorovat tyto procesy z každé instance. V tomto případě chci získávat metriky, jako je například stav procesu a zjišťovat, jestli

Obrázek 5.6: Ukázka dashboardu pro stav CEPH clusteru

je běžící a také například na jaké konkrétní instanci proces běží. Dále je třeba zaznamenat časový údaj pro aktuálnost informací.

Po vytvoření potřebných tabulek je také nutno vytvořit dva uživatele. Z důvodu bezpečnosti bylo nutné vytvořit uživatele, kteří mají omezená oprávnění. Pro uživatele, který do databáze zapisuje jednotlivé metriky, byla zpřístupněna konkrétní databáze a uživateli bylo přidáno oprávnění pro zápis. Druhým uživatelem je uživatel, který pouze z konkrétní databáze čte požadované metriky, což je uživatel určený pro grafanu a propojení s databází. Po dokončení všech těchto procesů jsem musel vytvořit script pro sběr potřebných dat ze systému a pro následné odesílání těchto dat do vytvořené databáze a jejich tabulek. Pro vytvoření tohoto scriptu jsem využil Powershell.

Ve scriptu využívám funkce jako Get-RemoteAccessConnectionStatistics pro sběr statistik o aktuálních VPN připojeních a funkci Get-Service s paramerty RemoteAccess a RasMan pro informace o konkrétních procesech. Tyto data poté odesílám pomocí Invoke-SQLcmd příkazu do databáze, kterou jsem si vytvořil. Tento script je nastaven pomocí nástroje Task Scheduler, aby se spouštěl co 5 minut a tím aktualizoval stávající data.

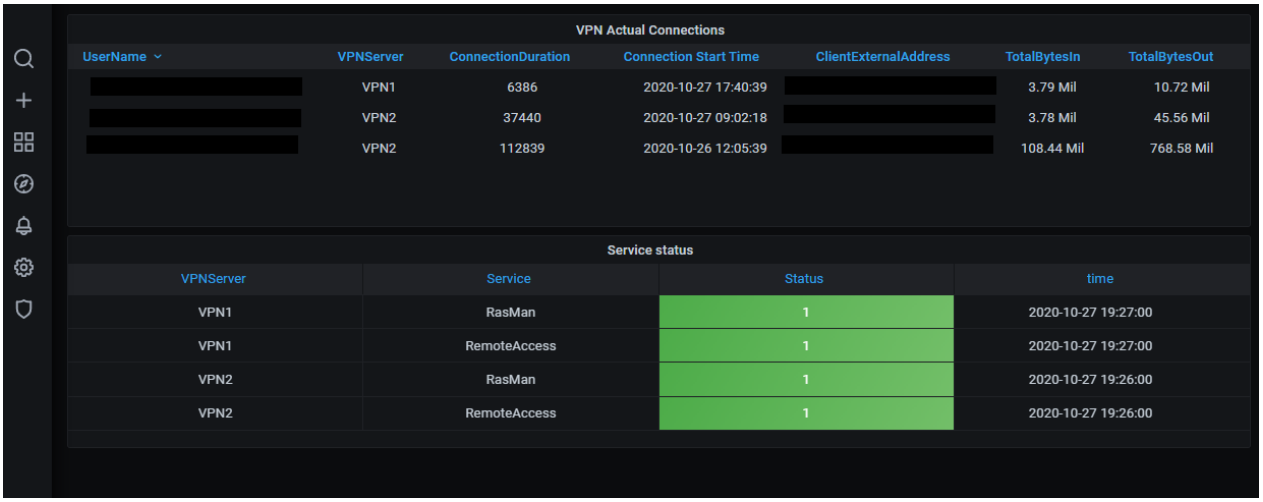
Po vytvoření scriptu a jeho spuštění se nyní do databáze nahrávají data. Tato data jsou ovšem pouze ve formě tabulky v databázi. Pro přehled těchto dat a jejich monitoring v konkrétním systému je nutno tyto data vizualizovat a připojit do Grafany. V Grafaně jsem přidal nový datový zdroj jako Microsoft SQL Server.

Po přidání datového zdroje je možno již pracovat s databází a z prostředí Grafany vytvářet

SQL dotazy a data vizualizovat. Pomocí dotazů na databázi jsem vytvořil konkrétní tabulkový dashboard, ve kterých jsou vidět jak informace o aktuálně přihlášených uživatelích, tak informace o konkrétních službách. Příklady dotazů a ukázka vytvořeného dasboardu je vidět níže.

```
1. SELECT VPNServer, Service, Status, Time FROM dbo.vpn_services_status

2. SELECT UserName, VPNServer, ConnectionDuration, ConnectionStartTime,
    ClientExternalAddress, TotalBytesIn, TotalBytesOut
FROM dbo.actual_connections
```



Obrázek 5.7: Ukázka vytvořeného dashboardu pro monitoring VPN v prostředí Grafana

Jak je možné vidět na obrázku 5.7, využil jsem tabulkového rozložení pro zobrazení konkrétních informací o procesech a o připojení uživatelů. Díky možnosti grafické vizualizace a mapování hodnot je možno vytvářet graficky přívětivé panely. V mém případě jsem využil právě mapování textu na hodnotu, kdy pro proces ve stavu Running je hodnota 1 a pro stav Stopped je hodnota 2. Pokud by nastal stav procesu jako Stopped, automaticky by se barva v panelu přebarvila na červenou a vizuálně by barvou označila chybu.

5.2.3 InfluxDB

Dalším z datových zdrojů, který jsem se rozhodl využít, je InfluxDB. InfluxDB je open-source time-series databáze, která umožňuje uchovávat time-series data a metriky v čase. Tuto databázi jsem se rozhodl využít pro sběr metrik ze systému PfSense, o kterém jsem se zmínil v sekci 3.4.1. Ze systému pfSense je třeba získávat data jako například vytížení systému, data o síťových rozhraních nebo třeba počet přihlášených uživatelů. PfSense podporuje plugin, který se nazývá Telegraf. Jedná se o plugin, který se chová na serveru jako agent a provádí sběr metrik ze systému. Umožňuje uložení těchto dat do různých databází. Více o tomto pluginu je možné najít na stránkách dokumentace viz [26].

V prostředí PfSense bylo tedy nutno nainstalovat Telegraf plugin, který následně umožňuje odesílat tyto metriky do externí databáze. Instalace se provádí pomocí volby Package Manager v záložce System. Plugin lze pak nalézt v záložce Available Packages a po nainstalování se objeví v Installed Packages, kde jsou všechny nainstalované pluginy v prostředí PfSense. Podrobná konfigurace je popsána na konci práce viz C.

Po nainstalování pluginu bylo nutno nastavit externí databázi, do které bude metriky plugin odesílat. Z tohoto důvodu jsem si vytvořil instanci, na které jsem vytvořil InfluxDB databázi. Pro instalaci InfluxDB je nutno opět přidat repozitář. Po přidání repozitáře je již možno provést instalaci pomocí příkazu yum. Po nainstalování je možno službu spustit. Dále je dobré zkontrolovat, zda je otevřený port 8086, na který se následně data odesílají.

Po spuštění je možno dostat se do administrace databáze. Pro vstup se využívá příkaz influx. Následně je nutno vytvořit databázi pro uchovávání dat a opět jako u MSSQL dva uživatele. Jeden uživatel je pro čtení dat z databáze a propojení s Grafanou a druhý uživatel je pro zápis dat z PfSense do databáze.

Po vytvoření databáze a uživatelů je možné provázat PfSense pomocí Telegraf pluginu s databází. Konkrétní nastavení se provádí pomocí volby pluginu Telegraf v záložce Services v prostředí PfSense. Je zde nutno zvolit databázi InfluxDB, dále nastavit IP adresu serveru, název databáze a uživatelské jméno a heslo pro zápis do databáze.

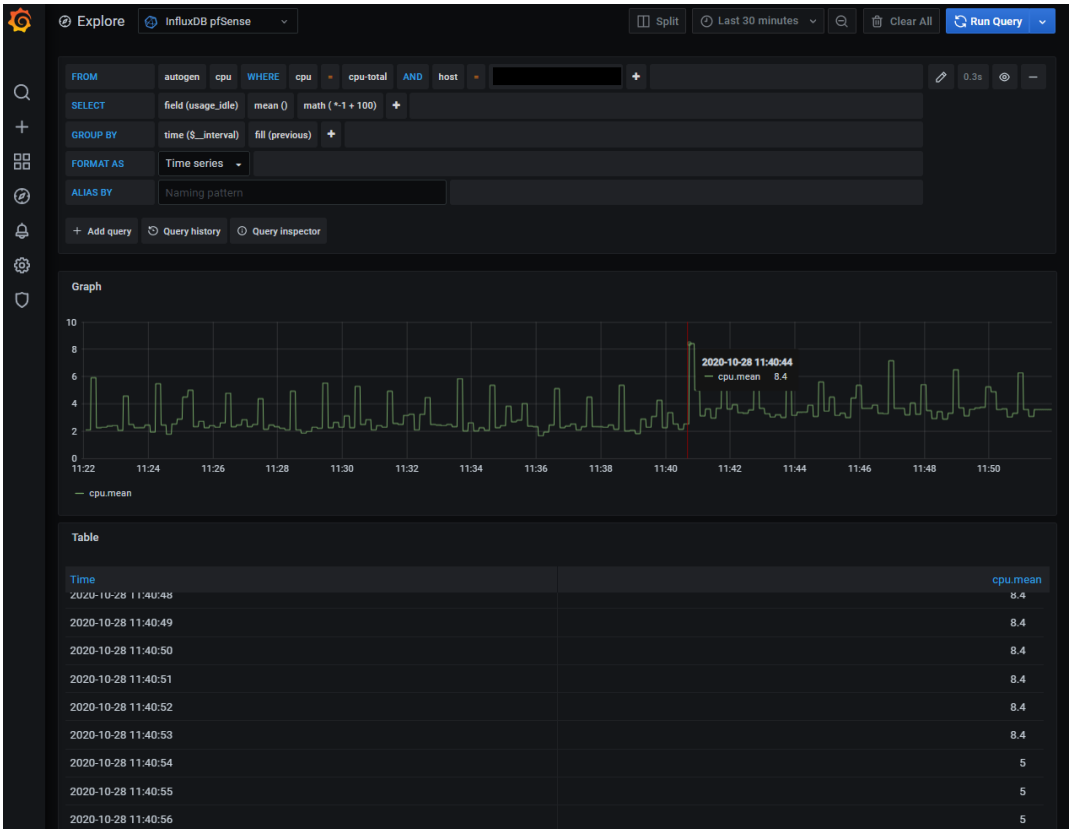
Pokud vše proběhlo správně je možno již zobrazit obsažené metriky v databázi. To lze provést pomocí příkazu SHOW MEASUREMENTS viz 5.8. Pokud jsou zde nějaké metriky je možno následně je zobrazit v Grafaně. Pro připojení InfluxDB do Grafany jsem opět využil přidání datového zdroje.

```
> SHOW MEASUREMENTS
name: measurements
name
----
cpu
disk
diskio
mem
net
pf
processes
swap
system
```

Obrázek 5.8: InfluxDB metriky z PfSense

Po připojení je již možno pracovat s metriky z databáze a vytvářet konkrétní dashboardy pro monitoring systému PfSense. Každý datový zdroj má většinou odlišnou syntaxi a možnosti jak s daty pracovat. Díky možnostem Grafany tuto syntaxi vždy na základě každého datového zdroje

identifikuje. Pro práci s InfluxDB se využívá InfluxDB query editor. Pro příklad jsem si vypsal vytížení celkového stavu CPU systému PfSense. Je zde možné pracovat s nejrůznějšími daty jako jsou síťová data nebo třeba systémová. Protože data jsou ve formě time series je možné je zobrazit v grafu na časové ose. Příklad takového dotazu je vidět na obrázku 5.9. Pomocí takových dotazů je následně možno tvořit různé dashboardy a vizualizace dat. Pro stav CPU, který jsem využil ve výsledném dashboardu, jsem si vytvořil panel, který obsahuje informace v procentech a stav jeho využití je označen barvou na základě definovaných thresholdů. Díky tomu bylo možno lépe vizualizovat konkrétní data a tím zpřehlednit informace o systému. Tento panel je vidět níže na obrázku 5.10.



Obrázek 5.9: Ukázka dotazu na stav CPU pro InfluxDB

Pro vytvoření konkrétního dashboardu pro PfSense jsem využil více panelů a různých typů metrik pro systémové informace a pro metriky síťových rozhraní. Těchto metrik je velké množství a popis všech konkrétních by bylo nad rámec této práce. Protože v infrastruktuře pro pfSense využíváme více cpu, více disků a různé síťové rozhraní využil jsem také možnosti vytvoření proměnných, na základě kterých je pak možno ve výsledném dashboardu filtrovat výsledky například pro konkrétní CPU nebo síťové rozhraní. K této konfiguraci je možno se dostat přes tlačítko Dashboard Settings a následně volby Variables. Zde je pak možno vytvořit si konkrétní proměnnou. To se pro-

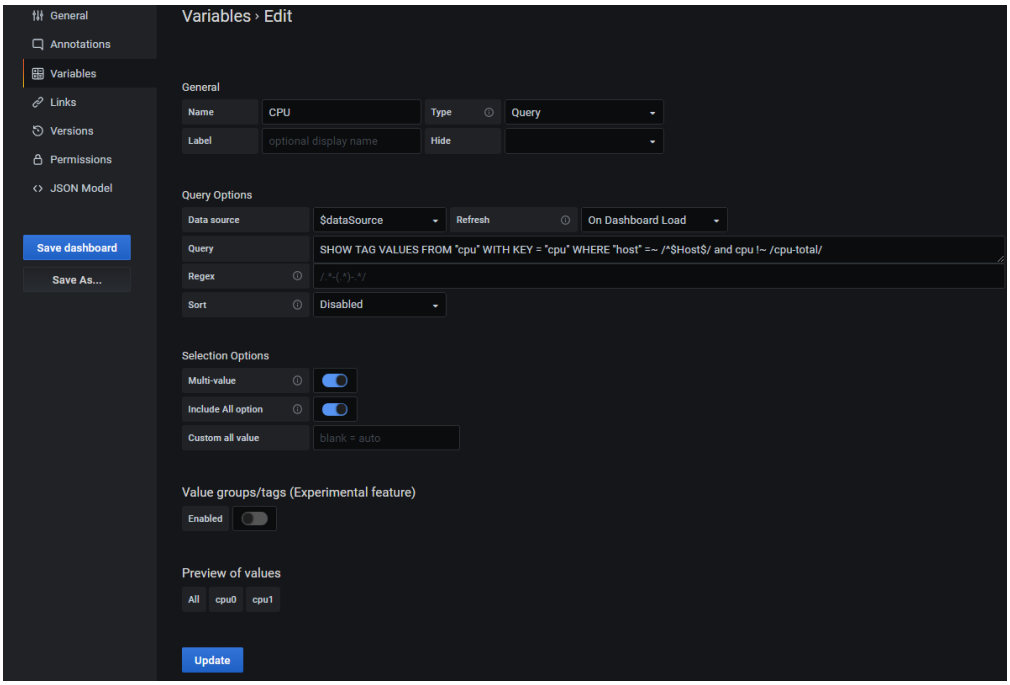
Obrázek 5.10: Ukázka dotazu na stav CPU v Gauge panelu.

vede pomocí dotazu, kdy je možné využít i různé typy regexů pro získání požadovaných výsledků. Pro příklad jsem si vybral již zmíněné CPU. V editoru je možné také vybrat prvky jako Multi-Value, což umožňuje vybrat více prvků najednou a také možnost Include All option, která ve filtru automaticky zobrazí možnost vybrat všechny prvky. Příklad tvorby proměnné je vidět na obrázku 5.11. Výsledný dashboard pro monitoring systému pfSense je vidět níže na obrázku 5.12.

5.2.4 Elasticsearch

Dalším datovým zdrojem, který jsem do Grafany připojil je Elasticsearch. Tento nástroj je součástí ELK stacku. Jedná se o distribuovaný vyhledávací a analyzační nástroj. Je založen na Apache Lucene a využívá RESTové API. Nejedná se o relační databázi. Nástroj se řadí do NoSQL databází a je velmi rychlý a škálovatelný. Lze v něm vyhledávat na základě filtrů a pomocí fulltextového vyhledávání. Používá se například pro správu logů.

V mém případě využívám ElasticSearch ve spojení s nástrojem Cerebro viz [27] a Graylog viz [28]. Využívám jej pro zaznamenávání logů ve formátu SYSLOG, pro zobrazení statistik z pfSense pro firewall. SYSLOG je standard pro zaznamenávání zpráv a logů v určitém formátu z různých zařízení prostřednictvím sítě [29]. Dále jej také využívám pro zobrazení všeobecných logů z pfSense. Protože v rámci monitoringu je nutno sledovat i logy rozhodl jsem se proto využít těchto nástrojů. Na základě firewall logů pak monitoruji metriky jako například počet firewall eventů, provoz na firewallu



Obrázek 5.11: Ukázka tvorby proměnné v prostředí Grafana

na základě portu, nejpoužívanější porty a podobně. Všechny nástroje potřebné pro vizualizaci těchto logů jsem nasadil a aktuálně běží na jednom virtuálním stroji s 2 CPU, 6 GB RAM a 60 GB diskem.

Prvním krokem byla implementace samotných nástrojů. Nainstaloval jsem tedy Elasticsearch, Graylog a Cerebro. Graylog zde využívám pro parsování logů a předávání těchto logů do ElasticSearch. Cerebro zde využívám pro kontrolu nad ElasticSearch. Po nainstalování bylo nutno nastavit v Graylogu set indexů, na základě kterého se data poté ukládají a mapují v nástroji Elasticsearch. Graylog pro každý index generuje vlastní šablonu, kterou je možno si přizpůsobit a pro data vytvořit filtry. Tuto šablonu jsem upravil pro ElasticSearch v nástroji Cerebro. Přidávám zde atributy jako například časový údaj uložení logu, který následně využívám v Grafaně pro procházení dat v čase. Po nastavení setu indexů je možno si jej zobrazit v nástroji Cerebro po připojení k ElasticSearch. Protože je ElasticSearch i Cerebro na jednom stroji, je možno využít localhost adresu. Dalším krokem bylo nainstalování balíčků pro formát syslog, extraktory dat a vyhledávací tabulky v nástroji Graylog. Na základě těchto balíčků je pak možno využívat parsování dat z nástroje pfSense a provádět jejich správu a filtrování. Po nainstalování bylo nutno nastavit příchozí stream dat pro pfSense. Zde bylo nutno specifikovat, jaké konkrétně logy se mají do Graylogu dostávat. Dále bylo nutno vytvořit v Graylogu pipeline, pomocí které jsem upravil formát pro časový údaj, aby se dal použít v prostředí Grafany.

Po nastavení všech nástrojů bylo nutno nastavit již samotné odesílání logů ve formátu syslog v systému pfSense. Toto nastavení lze provést v záložce Status a pomocí volby System Logs a Settings. Zde bylo nutno specifikovat, kam se tyto logy mají odesílat a jaké konkrétní logy se mají

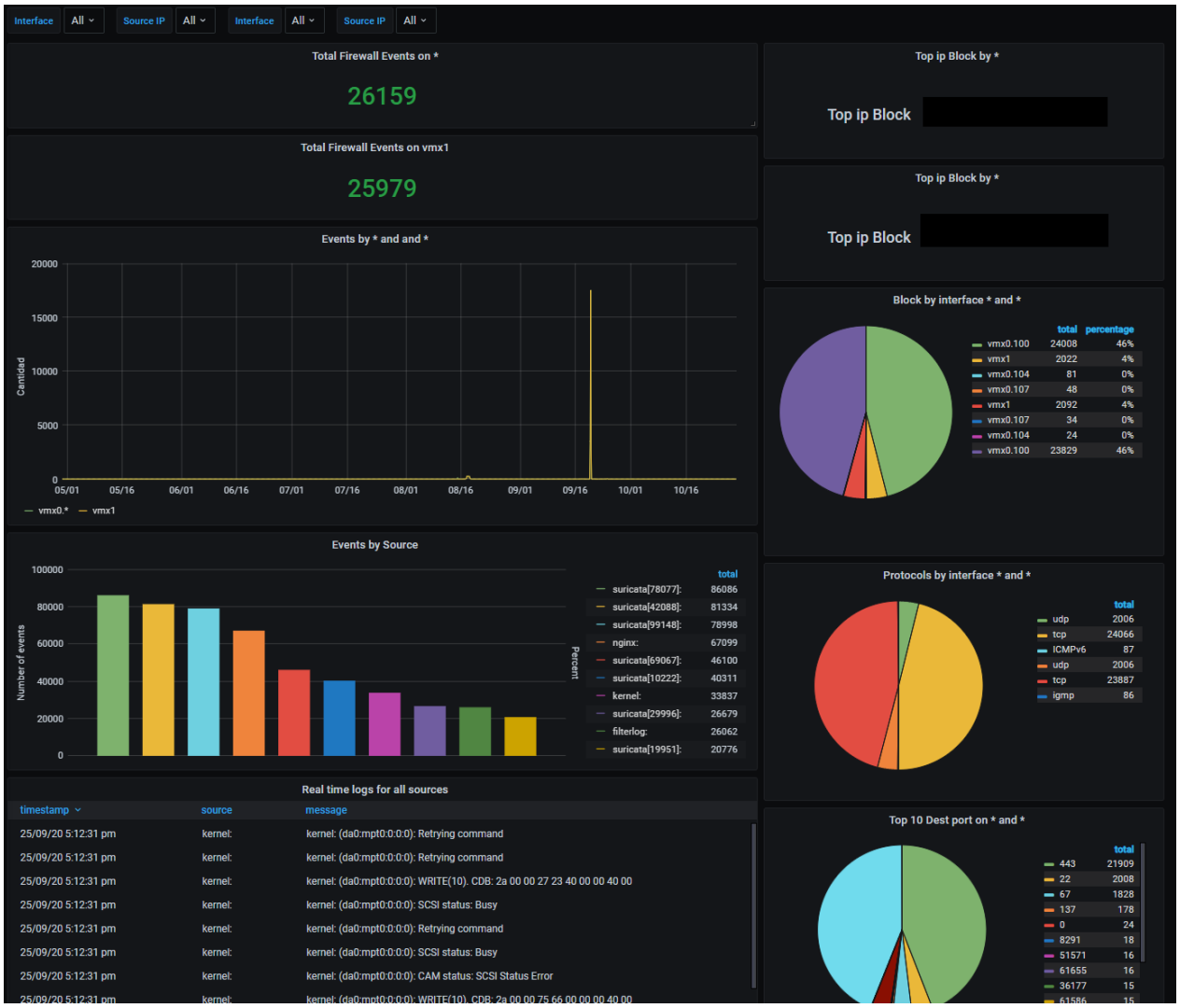
Obrázek 5.12: Ukázka výsledného dashboardu pro PfSense

odesílat. Protože mám zájem o všechny logy, vybral jsem tedy možnost pro odesílání všech logů. Po nastavení se již logy odesílají do Graylogu, kde se filtrují na základě šablon pro ElasticSearch nastavené prostřednictvím nástroje Cerebro. Pro veškeré nastavení doporučuji využít nejnovější verze nástrojů, jejich dokumentaci a při instalaci kontrolovat porty, na kterých nástroje fungují. V průběhu instalace jsem narazil na několik problémů jako jsou například duplicita portů, neotevřené porty na firewallu, nepříchozí logy a jejich filtrování kvůli špatně nastavených šablon v nástroji Cerebro a pravidel v nástroji pro filtrování v nástroji Graylog. Nakonec po nastavení jsem připojit ElasticSearch jako datový zdroj do Grafany. Při nastavování datového zdroje doporučuji dát pozor na správnou specifikaci setu indexů a atribut pro časový údaj.

Po úspěšném přidání datového zdroje bylo již možno vytvořit konkrétní dashboard pro vizualizaci logů a dat z firewallu. Finální dashboard pro vizualizaci firewall logů a všeobecných logů je možno vidět na obrázku níže viz 5.13.

5.2.5 Tvorba alertů v prostředí Grafana

V každém monitorovacím systému je třeba mít implementovaný systém alertů. Pokud by nastalo například spadnutí systému, nějaké instance, nebo například nedostupnost služeb, je nutné tento stav ohlásit, aby se mohl problém co nejrychleji řešit. Administrátoři pak nemusí neustále sledo-



Obrázek 5.13: Dashboard pro vizualizaci firewall a všeobecných logů z nástroje pfSense

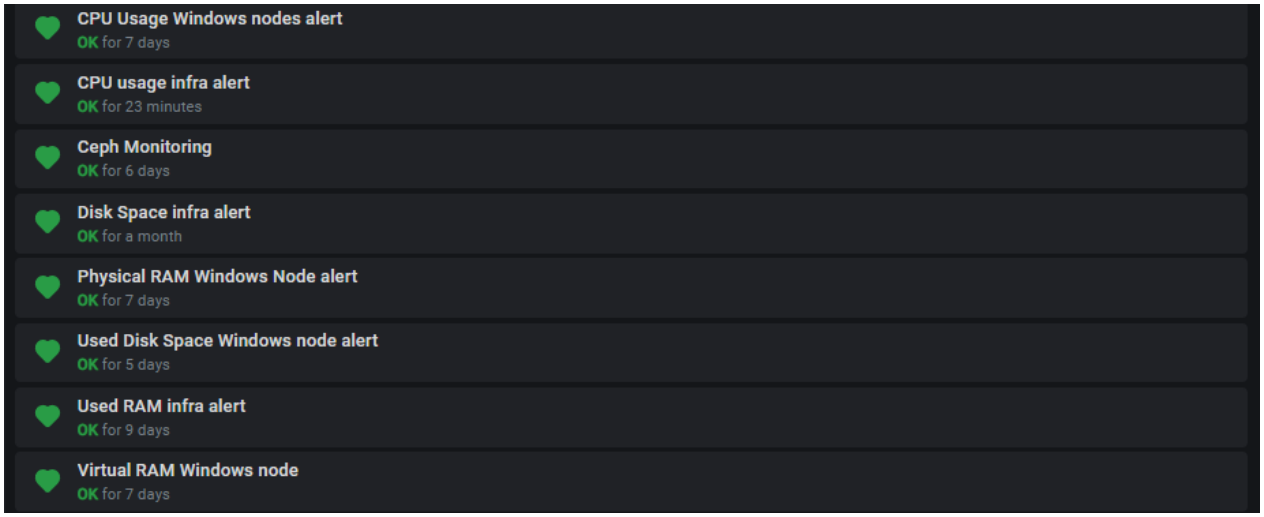
vat jednotlivé dashboardy, ale mohou se spolehnout na definovaná poplachová kritéria, při jejichž překročení dojde k odeslání oznámení o problému.

Pro implementaci alertingu využívám Grafanu. Grafana v základu obsahuje možnost vytváření alertů a nastavení thresholdů pro definovaná pravidla. Grafana má také možnost využití mnoha komunikačních kanálů. Je možné využít odesílání oznámení například přes Discord, Microsoft Teams, Telegram, Mattermost nebo Email. Grafana umožňuje tyto alerty vytvářet pro panel grafů.

Pro alerty jsem vytvořil dashboard, kde uchovávám panely grafů pro různé instance a různá kritéria pro alerting. Po vytvoření panelu grafu bylo třeba přidat dotazy na metriky, které potřebujeme monitorovat a na základě kterých se pak budou vytvářet thresholdy. Pro příklad jsem si vybral vytížení CPU na instancích s operačním systémem Windows. Pomocí dotazů do grafu vypisuji jednotlivé instance a jejich vytížení CPU. Následně pomocí volby Alert, vytvářím pro konkrétní dotazy

na instance pravidla, na základě kterých se odešle zpráva přes definované komunikační kanály. Jak je vidět na obrázku 5.15, pro každý dotaz mám definované pravidlo, při kterém se spustí hlášení.

Je možné definovat pro každý dotaz speciální threshold a tedy pro každou monitorovanou instanci vytvořit speciální pravidlo a limit pro spuštění. V tomto případě, pokud by vytížení CPU překročilo 85 procent, spustilo by se pravidlo pro hlášení. Toto hlášení se dostane do stavu Pending, kdy čeká 5 minut, zda hodnota vytížení neklesne pod 85 procent. Pokud by hodnota klesla, došlo by k přepnutí hlášení do stavu OK. Pokud ale hodnota i po 5 minutovém intervalu přesahuje definovaných 85 procent, dojde k přepnutí hlášení do stavu Firing a odešle se oznámení přes definované komunikační s informací o překročení stanoveného limitu. Jedním ze stavů je ještě stav No Data, který ohlašuje pokud by nebyly dostupné data dané instance. V mém případě alerty odesílám prostřednictvím kanálu Telegraf a Mattermost. Každý komunikační kanál má svůj způsob pro připojení. Telegram například požaduje vytvoření chatbota, který má k dispozici svůj API klíč, na základě kterého poté probíhá propojení. Na základě takto vytvořených alertů je pak možné vytvořit panel, například pro nějaký komplexní dashboard, a zobrazit si na něm stav jednotlivých vytvořených alertů. Na obrázku 5.14 je možné vidět vizualizaci alert panelu.



Obrázek 5.14: Příklad zobrazení alertů a jejich stavu ve formě panelu.

Obrázek 5.15: Příklad tvorby alertů pro vytížení CPU na instancích se systémem Windows

Kapitola 6

Záloha a obnovení zvoleného řešení

V této kapitole se zaměřuji na zálohu systémů a dat v rámci zvoleného řešení v případě, že by došlo k neopravitelné chybě, útoku, znepřístupnění systémů, případně by bylo třeba data, nastavení a celý systém obnovit. Otázka zálohování je ale poměrně složitá a je nutné se nad tímto tématem podrobněji zamyslet a uvědomit si, jak by vlastně taková záloha měla vypadat. Je třeba vzít v úvahu důležitost dat a jednotlivých logů. Některá méně potřebná data, nebo ta, která se nemění příliš často, stačí zálohovat třeba jednou týdně. Na druhou stranu kritická data nebo systémy by se měly zálohovat každý den. Další věcí, kterou je třeba určit je, jak dlouho zálohovaná data uchovávat a v jaké podobě. Také je nutno vzít v úvahu to, že jakékoliv ztráty dat v produkčním prostředí, popřípadě jejich nedostupnost mohou znamenat značné finanční ztráty. Z tohoto důvodu je také nutno v případě zálohy infrastruktury důkladně promyslet systém zálohování tak, aby byl spolehlivý, bezpečný a zaručoval uživatelům vždy možnost obnovy jejich dat v případě jakékoliv ztráty a aby toto obnovení bylo co nejrychlejší.

6.1 Typy zálohování

Existují různé typy zálohování jako jsou:

- Plná záloha

  Jedná se o nejzákladnější z typů záloh. V tomto případě se zálohují všechna data bez ohledu na to, kdy naposledy byla data změněna. Při obnově se následně obnoví všechna data, která byla zálohována v určitý čas.

- Inkrementální záloha

  V případě této zálohy dochází k zálohování pouze těch dat, které se změnily od poslední zálohy ať už se jednalo o inkrementální zálohu nebo plnou zálohu. U inkrementální zálohy jsou předchozí zálohy na sobě závislé.

- Diferenciální záloha

  Diferenciální záloha byla představena pro zjednodušení procesu zálohování a obnovy. Nabízí jednodušší obnovu dat než například inkrementální záloha. Jedná se o zálohu, která zálohuje všechna data, která se změnila od poslední plné zálohy. Na rozdíl od inkrementální zálohy nejsou na sobě zálohy závislé a je tedy možné předchozí diferenciální zálohy mazat.

- Záloha pomocí obrazu virtuálního stroje

  Záloha, která je nyní hojně využívaná v cloudových prostředích, kdy dochází k plné záloze konkrétního virtuálního stroje a následně aktualizace této zálohy na základě uživatelem definovaných časů zálohy. V případě obnovení z obrazu stroje je možné jej obnovit do přesného stavu, ve kterém byl, když proběhla záloha. Tyto obrazy je možné snadno uchovávat a pro lepší zabezpečení je také zašifrovat.

6.1.1 Zálohovací pravidlo 3-2-1

Zálohovací pravidlo 3-2-1 je pojem, který označuje doporučený způsob zálohování. Jedná se o zálohu, která obsahuje alespoň 3 kopie dat, která je třeba zálohovat. Tyto kopie jsou následně rozděleny na různá umístnění, aby v případě ztráty jedné kopie mohlo dojít k obnovení z jiných zdrojů, případně míst. Dvě kopie dat je doporučeno ukládat na různá externí média jako je například pevný disk nebo páska. Další kopie je dobré mít uložené úplně mimo pracoviště v případě, že by například došlo k vyhoření a podobně [30].

6.1.2 Snapshoty

Snapshoty jsou virtuální kopie disku. Jedná se o řešení, které umožňuje rychlou a jednoduchou obnovu poškozených či smazaných dat, případně celého systému. Díky snapshotům je možno také replikovat systém. Snapshoty využívají dvě metody, konkrétně Copy-on-write (COW) a Redirect-on-write (ROW) [31].

- Copy-on-write

  V případě COW je snapshot obsahující umístění bloků dat při vytvoření uložen do prostoru určeného pro snapshoty. Kdykoliv je následně nějaký původní blok nějak upraven je první nakopírován do prostoru pro snapshoty a až potom je původní blok upraven. Tento způsob zaručuje, že pokud by došlo k poškození dat během zápisu, tak se data obnoví z původního snapshotu. Nevýhoda metody COW je, že spotřebovává tři I/O operace, když je blok upraven. Jedná se o čtení původního bloku, poté zápis do separátního umístění a nakonec zápis upraveného bloku do původního umístění. Tím pádem, čím více dochází ke změně původních dat, tím častěji dochází k úpravě snapshotů a tím je vyšší spotřeba výkonu.

- Redirect-on-write

  V případě ROW, kdykoliv dojde k vytvoření snapshotu, tak se vytvoří kopie původního bloku a ukazatele na původní data jsou označeny jako read-only pro reprezentaci snapshotu. Ve chvíli, kdy jsou původní data změněna, jsou nová data přesměrována a zapsána do jiného místa. Původní mapa bloků je následně upravena tak, aby reflektovala nová data, zatímco ale snapshot ukazuje stále na data původní. V tomto případě se tedy redukuje počet operací potřebných k provedení ze tří v případě COW na jednu, a to zápis u ROW. Frekvence tvorby snapshotů může být tedy vyšší než u COW a také se mohou déle uchovávat.

6.2 Záloha monitorovacího systému

Monitorovací systém je jedním z hlavních prvků celé infrastruktury a je proto důležité jej zálohovat. Monitorovací systém využívá různé agenty a sbírá data a logy z odlišných lokalit a zdrojů a shromažďuje je na jednom místě v databázi na jednom stroji. Díky tomu je tedy možné využít zálohu celého stroje a není nutno zálohovat sbíraná data na straně agentů. Při použití více druhů zdrojů se samostatnými databázemi, je nutno myslet také na zálohu těchto konkrétních databází. V mém případě například zálohu databáze MySQL nebo InfluxDB. Další věcí, nad kterou je také dobré přemýšlet, je záloha konkrétních agentů a jejich nastavení v různých lokalitách. Je třeba zálohovat například vytvořené scripty v powershellu pro sběr dat. Samotné nasazené agenty není třeba zálohovat, pokud máme jejich konfiguraci kompletně uloženou v gitu. Tato konfigurace pak může být následně automaticky nasazována přes Ansible.

Implementace zálohy je prováděna dle pravidla 3-2-1. Kopie záloh jsou zajištěny pomocí Cephu. Ceph využívá Ceph pooly, které mohou být buď replikované, nebo je použití erasure coding. V základu je nastaven replikovaný typ poolu, díky kterému data z hlavního OSD disku kopíruje na jeden nebo více dalších disků. Tím zajišťuje, aby při selhání jednoho z disků byla data stále dostupná a dala se tedy obnovit. V případě využití erasure coding poolu, který je výpočetně složitější než například replikování, ale umožňuje snížit spotřebu místa na discích. Při tomto způsobu dochází k tomu, že se na ukládaný objekt v Ceph clusteru použije erasure coding, který objekt rozdělí na datové a kódové bloky a ty uloží na různé OSD disky [32].

6.2.1 Úrovně zálohy

Záloha samotné infrastruktury je rozdělena do více úrovní, aby byla všechna data bezpečně uložena a nedošlo k jejich ztrátě.

První úroveň je založena na exportu snapshotu RBD disků, což je funkcionalita CEPH clusteru. RBD umožňují sdílený fyzických zdrojů, lze měnit jejich velikost a ukládají data rozložené mezí OSD disky po celém Ceph clusteru. Poskytují tak distribuované a vysoce výkonné blokové úložné

disky viz [3]. Tyto snapshoty jsou tvořeny na denní bázi a následně jsou každý týden zálohovány na externí server.

Druhá úroveň zálohy je záloha v konkrétních cloudových řešeních. V našem případě se jedná o zálohu virtuálních strojů v prostředích VMware a Openstack. Tyto zálohy virtuálních strojů jsou prováděny denně. Ve VMware jsou pro zálohu využity snapshoty a v Openstacku je využita služba Cinder backup.

Třetí úrovní pro zálohu je záloha konkrétních služeb. Pro zálohu veškerých konfigurací jsou využity jak disky, tak cloudové řešení jako je GitLab. Všechny konfigurace je tedy možno v případě smazání obnovit a automatizovaně nasadit pomocí Ansible. Pro zálohu databází a služeb jsou implementovány různé metody, které se liší od služby. Jedná se například o zálohu struktury a dat SQL databáze pomocí textových souboru, které obsahují příkazy v jazyce SQL. V případě konfigurací pro monitoring jsou veškeré konfigurace pro monitoring server a agenty, kteří sbírají data v GitLabu a dále na záložních discích. Databáze, které jsou vytvořeny pro sběr dat od agentů, jako jsou již popsané MySQL a InnoDB, jsou tedy zálohované jak v případě zálohy celého virtuálního stroje, tak sql dumpů.

6.2.2 Obnova zvoleného řešení

Obnova infrastruktury a konkrétních systémů vychází ze zvoleného řešení zálohy. Obnova by měla být v případě výpadku rychlá a nemělo by dojít k poškození dat. V případě, kdy by došlo k selhání nějakého disku, vyhoření serveru, je možné na základě zálohy v Cephu pomocí rbd snapshotů jednotlivé disky obnovit. Tyto snapshoty, které se ukládají na externí server, je možné následně naimportovat zpátky do Ceph clusteru jako nový disk.

Tento disk může být následně naimportován do VMware jako další datastore a VMware s ním může ihned pracovat a tím také obnovit všechny virtuální stroje. V případě, že by nastala chyba například pouze v jednom virtuálním stroji v prostředí VMware, kdy by virtuální stroj nebyl spustitelný nebo by došlo ke smazání dat, je možné díky denní zálohy pomocí VMware snapshotů daný obraz virtuálního stroje obnovit a vrátit ho tak to původního stavu ještě než došlo k dané chybě. To samé platí pro Openstack, kdy by bylo možné využít importu zálohy.

V případě obnovení konfigurace a služeb je toto obnovení závislé na typu služby. V případě monitoring serveru by se jednalo například o obnovení konfigurace Promethea a Grafany, pro připojení k LDAP apod. je možné využít ansible k automatické konfiguraci služeb díky záloze a automatizaci zvoleného řešení. Pro databáze je možné použít import zálohy vytvořené pomocí sql dumpů.

Kapitola 7

Závěr

V rámci této práce jsem získal mnoho zkušeností s návrhem infrastruktury, její tvorbou a s hledáním nejvhodnějších řešení, která odpovídají nejnovějším technologiím a standardům. Všechny tyto zkušenosti jsem využil díky mé práci na mezinárodním H2020 projektu LEXIS, na jehož řešení se podílím v rámci své práce na IT4Innovations. Hlavním výsledkem mé práce je komplexní monitorovací systém využitý v praxi, který je složen z různých druhů technologií, měl jsem tedy možnost využít široké znalosti z různých oblastí, které jsem získal za celou dobu studia. Jedná se o různé znalosti jako jsou správa databází, automatizace, skriptování nebo obsluha serverů a různých operačních systémů, ať už se jedná o UNIXové systémy, či Windows. Prošel jsem fázemi návrhu infrastruktury, její optimalizací a její konkrétní realizací. Díky vytvořeného monitorovacího systému je možné sledovat různé úrovně infrastruktury a její služby a tím tak předcházet chybám a eliminovat případné výpadky na co nejmenší dobu. Všechna tato data z různých zdrojů je možno sledovat v jednom prostředí. Setkal jsem se také s novými technologiemi jako je Ceph, nebo Openstack a díky této práci jsem také získal široký rozhled v rámci monitorování infrastruktury a jednotlivých služeb. Seznámil jsem se s řešením zálohy infrastruktury a jejich nejvhodnějších řešení.

V průběhu nasazování některých služeb, automatizace nebo také tvorby dashboardů pro datové zdroje jsem narazil na nespočet problémů, které vznikly například odlišnými verzemi, špatnou syntaxí nebo různými datovými zdroji. Tyto problémy jsem byl schopen úspěšně vyřešit. Návrh, realizace a optimalizace infrastrukutry a jejích částí je časově náročná záležitost, která vyžaduje neustálé učení, hledání nových a lepších řešení, která se vyskytnou a vyžaduje rozhled ve velké škále technologií.

Do budoucna plánuji vylepšovat možnosti monitoringu pro infrastrukturu se stále vyvýjejícími se funkcemi pro monitorovací systém a s rozšířením o další prvky pro monitorování infrastruktury. Dále také plánuji pracovat na projektu LEXIS a pomáhat s optimalizací a tvorbou infrastruktury. S tím souvisí také další učení a rozšiřování mých obzorů v oblasti nejnovějších technologií, automatizací systémů, cloudových řešení a správy serverů.

Literatura

1. LIGUS, Slawek. *Effective monitoring and alerting*. "O'Reilly Media, Inc.", 2012.
2. SCIONTI, Alberto et al. HPC, Cloud and Big-Data Convergent Architectures: The LEXIS Approach. In: BAROLLI, Leonard; HUSSAIN, Farookh Khadeer; IKEDA, Makoto (ed.). *Complex, Intelligent, and Software Intensive Systems*. Cham: Springer International Publishing, 2020, s. 200–212. ISBN 978-3-030-22354-0.
3. SINGH, Karan. *Learning Ceph*. Packt Publishing Ltd, 2015.
4. MARSHALL, Nick; BROWN, Mike; FRITZ, G Blair; JOHNSON, Ryan. *Mastering VMware VSphere 6.7*. John Wiley & Sons, 2018.
5. *VMware vSphere Documentation*. Dostupné také z: `https://docs.vmware.com/en/VMware-vSphere/index.html`.
6. SHRIVASTWA, Alok; SARAT, Sunil; JACKSON, Kevin; BUNCH, Cody; SIGLER, Egle; CAMPBELL, Tony. *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, 2016.
7. *OpenStack dashboard*. Dostupné také z: `https://docs.openstack.org/openstackdocstheme/latest/demo/dashboard_demo.html`.
8. ZIENTARA, David. *Learn pfSense 2.4: Get up and running with Pfsense and all the core concepts to build firewall and routing solutions*. Packt Publishing Ltd, 2018.
9. FRANCIS, Dishan. *Mastering Active Directory: Deploy and Secure Infrastructures with Active Directory, Windows Server 2016, and PowerShell*. Packt Publishing Ltd, 2019.
10. RAMIREZ, Nick. *Load Balancing with HAProxy: Open-source technology for better scalability, redundancy and availability in your IT infrastructure*. Independently published, 2016.
11. GUPTA, Yuvraj. *Kibana essentials*. Packt Publishing Ltd, 2015.
12. *Kibana features*. Dostupné také z: `https://www.elastic.co/kibana/features`.
13. OLUPS, Rihards; DALLE VACCHE, Andrea; UYTTERHOEVEN, Patrik. *Zabbix: Enterprise Network Monitoring Made Easy*. Packt Publishing Ltd, 2017.
14. *Zabbix Documentation 5.0*. Dostupné také z: `https://www.zabbix.com/documentation/current/manual/introduction/about`.

15. KOCJAN, Wojciech. *Learning Nagios 4*. Packt Publishing Ltd, 2014.

16. *About Nagios Core*. Dostupné také z: https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/about.html.

17. *Nagios XI - Easy Network, Server Monitoring and Alerting*. Dostupné také z: https://www.nagios.com/products/nagios-xi/#pricing.

18. BALLESTRERO, S; BRASOLIN, F; DÂRLEA, G–L; DUMITRU, I; SCANNICCHIO, D A; TWOMEY, M S; VÂLSAN, M L; ZAYTSEV, A. Tools and strategies to monitor the ATLAS online computing farm. *Journal of Physics: Conference Series*. 2012, roč. 396, č. 4, s. 042053. Dostupné z DOI: 10.1088/1742-6596/396/4/042053.

19. *About Icinga 2*. Dostupné také z: https://icinga.com/docs/icinga2/latest/.

20. BRAZIL, Brian. *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring*. "O'Reilly Media, Inc.", 2018.

21. PROMETHEUS. *Overview*. Dostupné také z: https://prometheus.io/docs/introduction/overview/.

22. SALITURO, E. *Learn Grafana 7.0: A beginner's guide to getting well versed in analytics, interactive dashboards, and monitoring*. Packt Publishing, 2020. ISBN 9781838828318. Dostupné také z: https://books.google.cz/books?id=0XntDwAAQBAJ.

23. *Grafana Features*. Dostupné také z: https://grafana.com/grafana/.

24. *Install on RPM-based Linux (CentOS, Fedora, OpenSuse, Red Hat)*. Dostupné také z: https://grafana.com/docs/grafana/latest/installation/rpm/.

25. *Ansible Documentation¶*. 2020. Dostupné také z: https://docs.ansible.com/ansible/latest/index.html.

26. *Telegraf 1.14 documentation*. Dostupné také z: https://docs.influxdata.com/telegraf/v1.14/.

27. LMENEZES. *Cerebro*. Dostupné také z: https://github.com/lmenezes/cerebro.

28. *Industry Leading Log Management*. Dostupné také z: https://www.graylog.org/.

29. GERHARDS, Rainer. *The Syslog Protocol* [RFC 5424]. RFC Editor. Request for Comments, č. 5424. Dostupné z DOI: 10.17487/RFC5424.

30. CRUMP, George. *The 7 critical backup strategy best practices to keep data safe*. TechTarget, 2019. Dostupné také z: https://searchdatabackup.techtarget.com/feature/The-7-critical-backup-strategy-best-practices-to-keep-data-safe.

31. HALABI, S. *Hyperconverged Infrastructure Data Centers: Demystifying HCI*. Pearson Education, 2019. Networking Technology. ISBN 9780134997926.

32. *ERASURE CODE POOLS Red Hat Ceph Storage 1.3*. Dostupné také z: https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/1.3/html/storage_strategies_guide/erasure_code_pools.

Příloha A

Konfigurace VPN monitoringu

Tato sekce je určena pro prezentaci skriptů a tvorby databáze, jejichž proces je popsán v sekci 5.2.2.

Příkaz pro vytvoření tabulky pro uložení dat k zaznamenání informací o uživatelích.

```
CREATE TABLE [dbo].[actual_connections](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [VPNServer] [nvarchar](255) NOT NULL,
   [ClientIPAddress] [nvarchar](255) NOT NULL,
   [UserName] [nvarchar](max) NOT NULL,
   [ConnectionDuration] [int] NOT NULL,
   [ConnectionStartTime] [datetime] NOT NULL,
   [ClientExternalAddress] [nvarchar](255) NOT NULL,
   [AuthMethod] [nvarchar](255) NOT NULL,
   [TunnelType] [nvarchar](255) NOT NULL,
   [TotalBytesIn] [int] NOT NULL,
   [TotalBytesOut] [int] NOT NULL
)
```

Příklad vytvoření tabulky pro zaznamenávání stavu procesů.

```
CREATE TABLE [dbo].[vpn_services_status](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [VPNServer] [nvarchar](255) NOT NULL,
   [Service] [nvarchar](255) NOT NULL,
   [Status] [nvarchar](max) NOT NULL,
   [Time] [datetime] NOT NULL
)
```

Script v Powershellu pro sběr dat a jejich odesílání do databáze. Některé hodnoty jsou pro účely bezpečnosti zaměněny.

```
Invoke-Sqlcmd -ServerInstance 'hostname,port' -Query "DELETE FROM dbo.
    actual_connections where VPNServer like '$env:computername'" -U
    Nazev_uzivatele -P Heslo_uzivatele -Database nazev_databaze
Invoke-Sqlcmd -ServerInstance 'hostname,port' -Query "DELETE FROM dbo.
    vpn_services_status where VPNServer like '$env:computername'" -U
    Nazev_uzivatele -P Heslo_uzivatele -Database nazev_databaze

$remotes=Get-RemoteAccessConnectionStatistics

foreach($remote in $remotes)
{
$username=$remote.UserName
$connduration=$remote.ConnectionDuration
$starttime=$remote.ConnectionStartTime
$clientip=$remote.ClientIPAddress
$extip=$remote.ClientExternalAddress
$authmeth=$remote.AuthMethod
$tunnelt=$remote.TunnelType
$bytesin=$remote.TotalBytesIn
$bytesout=$remote.TotalBytesOut
$server=$env:computername

$datainsertion="
INSERT INTO [dbo].[actual_connections]
          ([UserName]
          ,[ConnectionDuration]
          ,[ConnectionStartTime]
          ,[ClientIPAddress]
          ,[ClientExternalAddress]
          ,[AuthMethod]
          ,[TunnelType]
          ,[TotalBytesIn]
          ,[TotalBytesOut]
          ,[VPNServer])
     VALUES
          ('$username'
```

```
          ,'$connduration'
          ,'$starttime'
          ,'$clientip'
          ,'$extip'
          ,'$authmeth'
          ,'$tunnelt'
          ,'$bytesin'
          ,'$bytesout'
          ,'$server')
GO
"
Invoke-SQLcmd -ServerInstance 'hostname,port' -query $datainsertion -U
    Nazev_uzivatele -P Heslo_uzivatele -Database nazev_databaze

}

$vpn_status=Get-Service "RemoteAccess", "RasMan"
foreach($status in $vpn_status)
{
$vpnstat=$status.Status
$servicename=$status.name
$vpnserver=$env:computername
$time = Get-Date -UFormat "%m/%d/%Y %R"

$datains="
INSERT INTO [dbo].[vpn_services_status]
          ([Status]
          ,[Service]
          ,[VPNServer]
          ,[Time])
     VALUES
          ('$vpnstat'
          ,'$servicename'
          ,'$vpnserver'
          ,'$time'
        )
GO
"
```

```
Invoke-SQLcmd -ServerInstance 'hostname,port' -query $datains -U Nazev_uzivatele -
    P Heslo_uzivatele -Database nazev_databaze

}
```

Příloha B

Grafana konfigurace

V této sekci jsou popsány konfigurační soubory pro nástroj Grafana a jeho instalace, jehož proces je popsán v sekci 5.1.1.

První instalace nástroje Grafana na systému CentOS7.

```
1. #Vytvoření souboru pro repozitář
sudo vim /etc/yum.repos.d/grafana.repo

2. #Přidání informací pro instalační balíček nástroje Grafana
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

Po vytvoření tohoto souboru je již možné nástroj Grafana nainstalovat. To se provede pomocí příkazu

```
sudo yum install grafana
```

Po nainstalování balíčku provádím spuštění procesu, kontrolu zda naběhl v pořádku a nastavení automatického spuštění procesu po startu systému.

```
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

```
sudo systemctl enable grafana-server
```

Ověření otevřeného portu 3000.

```
firewall-cmd --zone=public --add-port=3000/tcp --permanent
```

Konfigurace souboru /etc/grafana/grafana.ini

```
[auth.ldap]
# Povolení ldap propojení
enabled = true

# Konfigurační soubor pro nastavení skupin a propojení s AD
config\_file = /etc/grafana/ldap.toml

# Povolení uživatelům přihlášení do nástroje a vytvoření uživatelského profilu po
    úspěšné autentifikaci.
allow\_sign\_up = true
```

Níže je příklad vytvořeného konfiguračního souboru a popis atributů, které je třeba upravit. V ukázce popisuju informativně jednotlivé bloky. V případě popisu každého atributu doporučuji využít dokumentace viz [24].

```
1. sudo vim /etc/grafana/ldap.toml

2. #Úprava souboru ldap.toml

[[servers]]
# IP adresa LDAP serveru proti kterému se bude ověřovat
host = "IP Adresa"
# Defaultní port, který bude využit. V případě využití ssl je a ověřování pomocí
    certifikátu je dobré využít port 636 jinak 389 v případu bez ssl. V mém případ
    ě využívám port 636. Dále je zde nastavení cest k certifikátům a možnosti SSL
    ověření.
port = 636
use_ssl = true
start_tls = false
ssl_skip_verify = false
root_ca_cert = "/path/to/certificate.crt"
client_cert = "/path/to/client.crt"
client_key = "/path/to/client.key"
```

```
# Pro čtení dat z AD je nutné si vytvořit uživatele, který se následně použije v
    konfiguraci.
bind_dn = "jméno_uživatele@doména"
bind_password = 'heslo_uživatele'

# Filtr na základě kterého bude probíhat vyhledávání v AD. sAMAccountName označuje
     v AD přihlašovací jméno uživatele.
search_filter = "(sAMAccountName=%s)"

# Doména, ve které se vyhledává.
search_base_dns = ["dc=corp,dc=local"]

# Specifikování názvů atributů, které v AD využíváme a jejich namapování na
    atributy Grafany.
[servers.attributes]
name = "givenName"
surname = "sn"
username = "sAMAccountName"
member_of = "memberOf"
email = "mail"

# Zde je jedna velmi důležitá skupina v konfiguraci, kde se definují uživatelské
    role. Grafana má role jako Admin, Editor a Viewer. Tyto role označují možnosti
     uživatele pracovat se systémem. Díky tomuto namapování na skupiny v AD je mož
    né následně kontrolovat uživatelská oprávnění na základě skupin.

#Příklad namapování skupiny pro administrátory
[[servers.group_mappings]]
group_dn = "cn=grafana-administrator,ou=projekt,dc=corp,dc=local"
org_role = "Admin"
grafana_admin = true

#Příklad namapování skupiny pro editory
[[servers.group_mappings]]
group_dn = "cn=grafana-edit,ou=projekt,dc=corp,dc=local"
org_role = "Editor"
```

```
#Příklad namapování skupiny pro pouze možnost sledovat jednotlivé dashboard ale
    bez možnosti jejich úpravy.
[[servers.group_mappings]]
group_dn = "cn=grafana-read,ou=projekt,dc=corp,dc=local"
org_role = "Viewer"
```

V sekci 5.1.1 jsem také zmiňoval automatické nasazení pomocí ansible. Díky automatizaci je možné poté proces konfigurace popsaný výše zjednodušit a tím si tak ušetřit čas s danou konfigurací a specifikováním atributů. Pro příklad je níže vidět výňatek z automatizačního skriptu pro nastavení LDAP připojení.

```
grafana_ldap:
  servers:
    host: IP Adresa
    port: 636
    use_ssl: true
    start_tls: false
    ssl_skip_verify: false
    bind_dn: "jméno_uživatele@doména"
    bind_password: "heslo_uživatele"
    search_filter: "(sAMAccountName=%s)"
    search_base_dns:
      -  "dc=corp,dc=local"
    attributes:
      name: givenName
      surname: sn
      username: sAMAccountName
      member_of: memberOf
      email: mail
    group_mappings:
      - name: skupina
        id: 1
        groups:
          - group_dn: "cn=grafana-administrator,ou=projekt,dc=corp,dc=local"
            org_role: Admin
            grafana_admin: true
          - group_dn: "cn=grafana-edit,ou=projekt,dc=corp,dc=local"
            org_role: Editor
          - group_dn: "cn=grafana-read,ou=projekt,dc=corp,dc=local"
```

```
        org_role: Viewer
```

Příloha C

InfluxDB a pfSense konfigurace

V této sekci je popsána konfigurace a ukázky pro sběr dat z prostředí pfSense, jehož nastavení a proces jsem popsal v sekci 5.2.3.

Ukázka instalace pluginů v uživatelském rozhraní prostředí PfSense je vidět na obrázcích níže viz C.1.

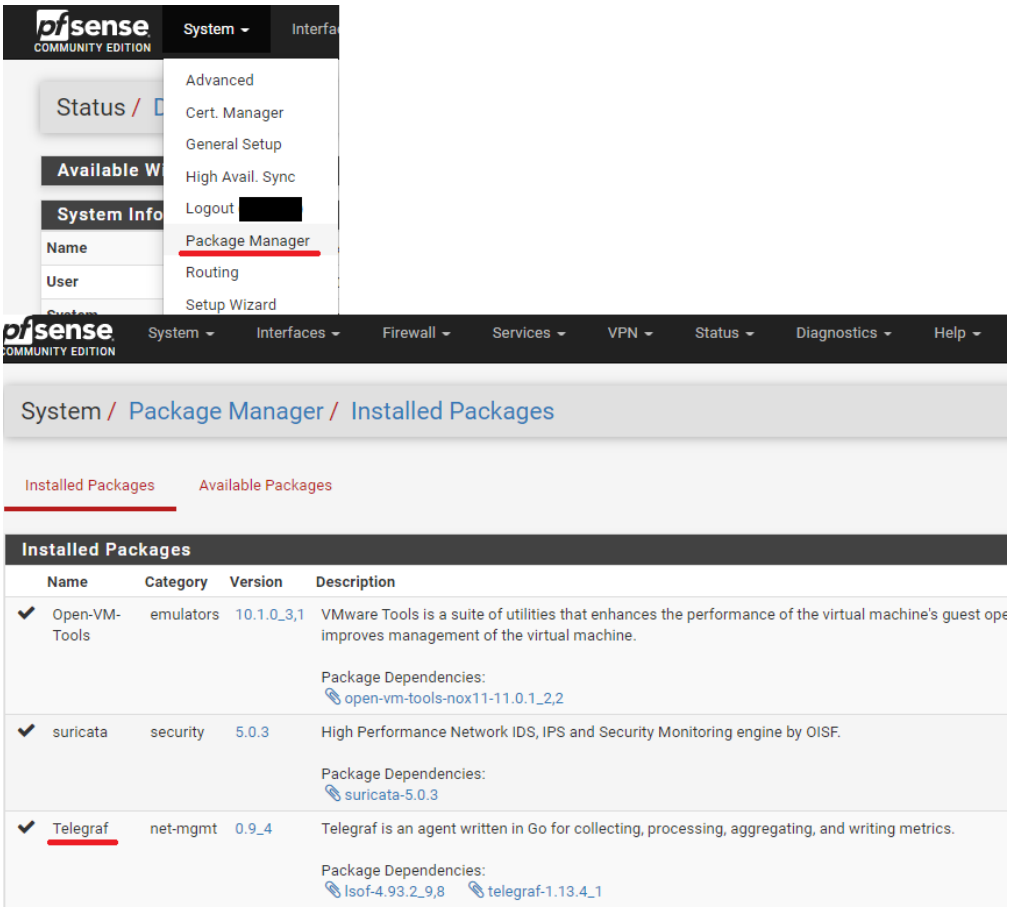
Proces instalace InfluxDB je popsán pomocí příkazu níže.

```
1. cat <<EOF | sudo tee /etc/yum.repos.d/influxdb.repo
[influxdb]
name = InfluxDB Repository - RHEL \$releasever
baseurl = https://repos.influxdata.com/rhel/\$releasever/\$basearch/stable
enabled = 1
gpgcheck = 1
gpgkey = https://repos.influxdata.com/influxdb.key
EOF

2. sudo yum install influxdb
3. sudo systemctl start influxdb
4. sudo systemctl enable influxdb
```

Proces vytvoření databáze, přidání uživatelů a uživatelských oprávnění je vidět níže.

```
1. influx
2. CREATE DATABASE "pfsense";
3. use pfsense
4. CREATE USER "pfsenseWriter" WITH PASSWORD 'heslo_pro_uzivatele';
5. CREATE USER "pfsenseReader" WITH PASSWORD 'heslo_pro_uzivatele';
6. GRANT READ ON pfsense TO pfsenseReader
7. GRANT WRITE ON pfsense TO pfsenseWriter
```

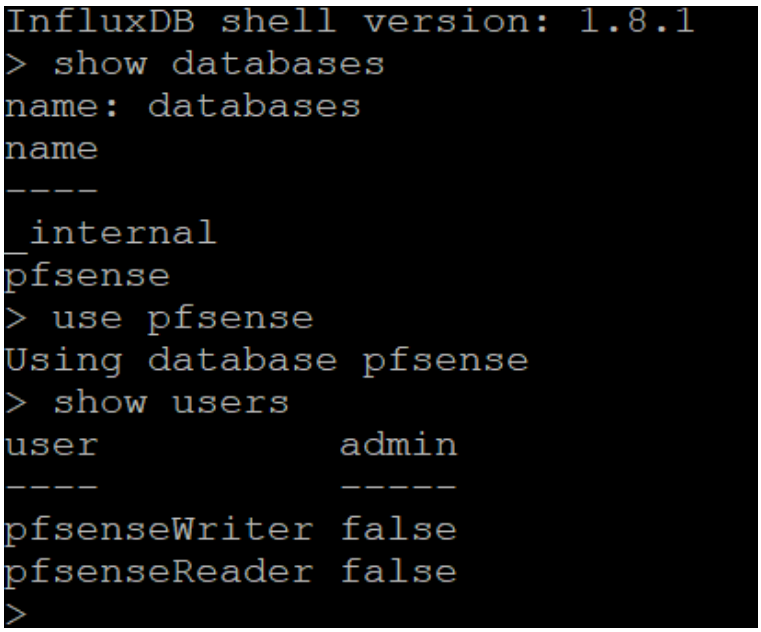


Obrázek C.1: Instalace Telegraf pluginu v prostředí PfSense

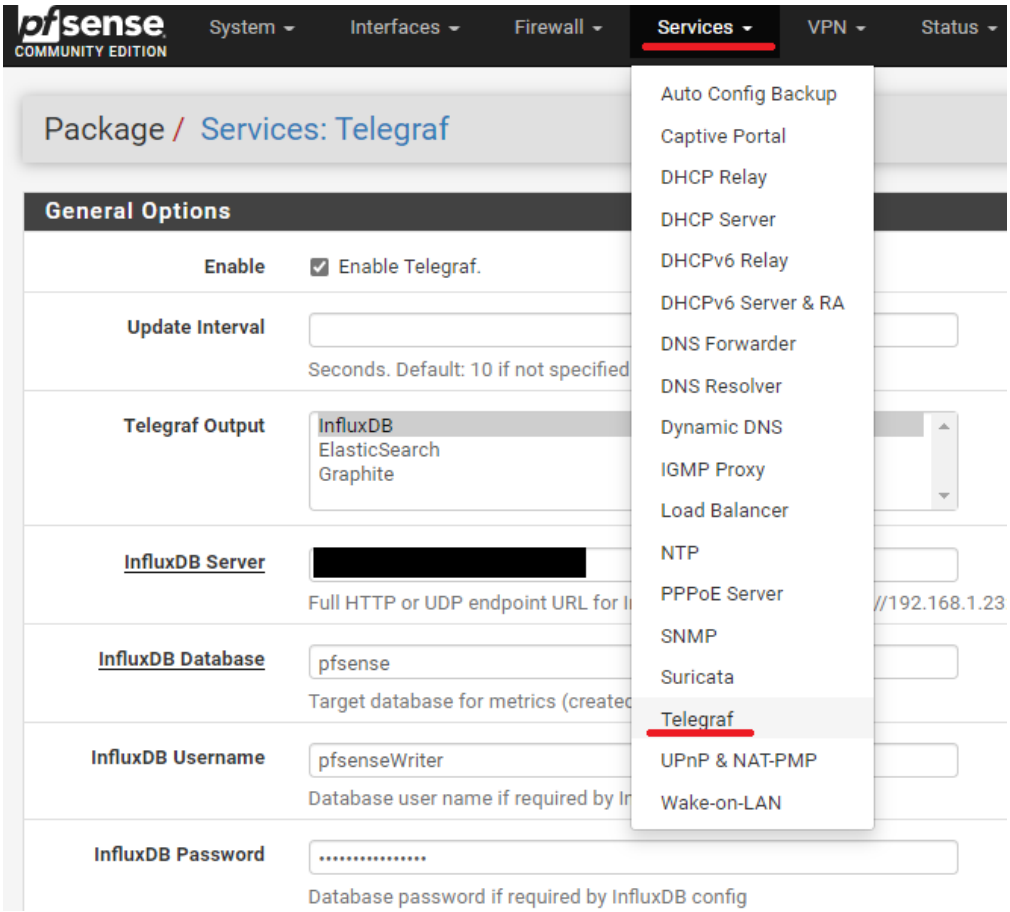
Pro ověření zda je databáze a její uživatelé vytvořeni, je dobré si vše vypsat. Výpis je vidět na obrázku C.2.

Nastavení databáze InfluxDB a propojení v prostředí pfSense. V mém případě je to uživatel pfsenseWriter. Nastavení je znázorněno na obrázku C.3.

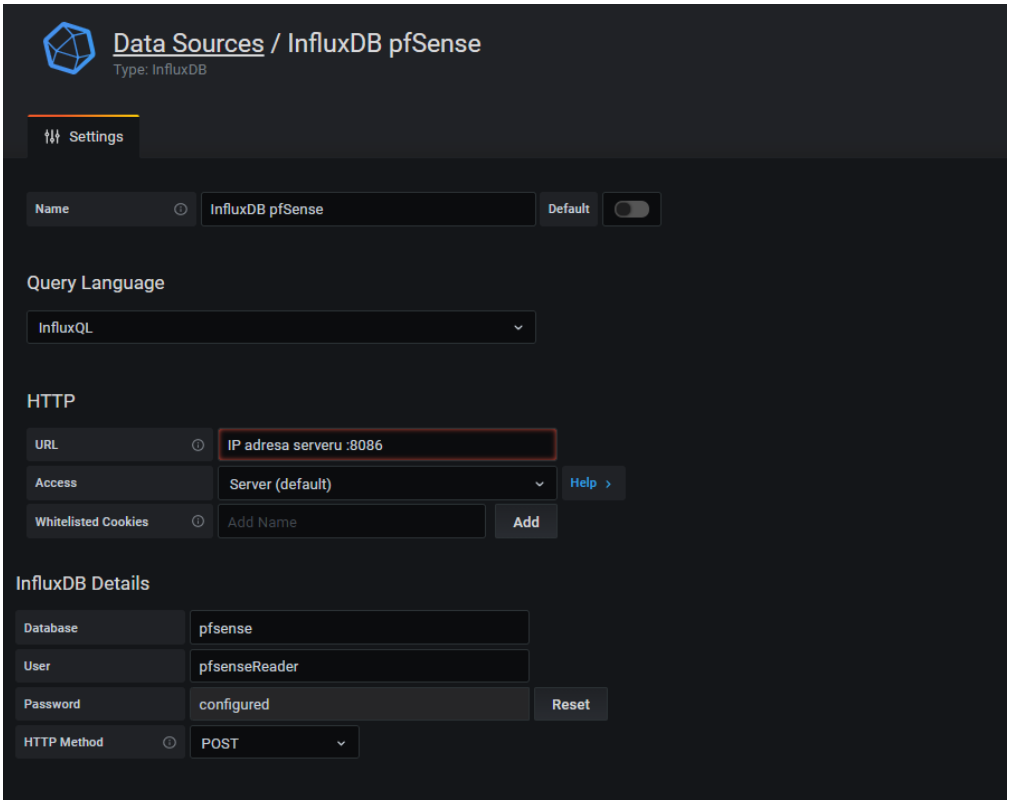
Pro připojení InfluxDB do Grafany jsem opět využil přidání datového zdroje. Nyní jsem vybral InfluxDB. Následně bylo třeba nakonfigurovat nastavení pro připojení k databázi. Jedná se konkrétně o IP adresu serveru, na kterém běží InfluxDB a poté nastavení údajů pro vytvořeného uživatele s přístupy ke čtení z databáze. Nastavení je vidět na obrázku C.4.



Obrázek C.2: Výpis databáze a uživatelů z InfluxDB



Obrázek C.3: Propojení Telegraf pluginu s InfluxDB



Obrázek C.4: Připojení InfluxDB do Grafany